Performance Evaluation of Multiple TCP Circuits with the help of Semaphore Queues

by

PAWAN KUMAR

TH
EE/1999/M
K96p

Department of Electrical Engineering

INDIAN INSTITUTE OF TECHNOLOGY KANPUR

April, 1999

Performance Evaluation of Multiple TCP Circuits with the help of Semaphore Queues

A Thesis Submitted
in Partial Fulfillment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY
by
PAWAN KUMAR

to the
Department of Electrical Engineering
INDIAN INSTITUTE OF TECHNOLOGY KANPUR
APRIL, 1999

CERTIFICATE

This is to certify that the work contained in this thesis entitled ***Performance Evaluation of Multiple TCP Circuits with the help of Semaphore Queues*** by **Pawan Kumar** has been carried out under our supervision and that this work has not been submitted elsewhere for a degree.

[illegible]

for Dr. Sanjay K. Bose,
Professor
Department of Electrical Engineering,
Indian Institute of Technology, Kanpur

[illegible]

Dr. Y. N. Singh
Assistant Professor
Department of Electrical Engineering,
Indian Institute of Technology, Kanpur

Dedicated

to

My Dear father

ACKNOWLEDGEMENT

My heartily thanks to Dr. S K. Bose for giving me this problem and providing me excellent guidance during the course of this thesis work. I am grateful to Dr. S. K Bose for encouraging and supporting me throughout the thesis work During the entire project I have received endless help from Dr Y.N Singh for finalizing the writing of this thesis and method of presentation. He has also given me moral boost up whenever I feel some depression.

I must also thank Vineet, Naresh, Himanshu, and Anil who helped me a lot.

I would also like to thank Mr. Bhatnagar, Mr. Kaushal, Mr. T. Srinivas Rao, Mr. Vivek Mudgil, Mr. Vineet, for providing me various kinds of help in the lab.

Last but most important I am indebted to my parents, sister and brothers for sacrificing all their pleasure and being on myside all the time. Without their support I couldn't have achieve my goal.

PAWAN KUMAR

Contents

List of Figures

List of Tables

ABSTRACT

This thesis presents an approach for approximately analyzing the performance of multiple TCP connections over a network with partially shared links. In this kind of network, two or more TCP connections may be forced to share the bandwidth of a common link The number of packets for a TCP connection is modeled as being controlled through a semaphore queue whose performance depends on the window size of the connection. This window size will also control the traffic flow contributed by this connection to the overall network. The network can then be modeled as a multiple- class closed queuing network, which has then been solved using a Mean Value Analysis (MVA) approach We have used this approach to analyze the performance of a network with multiple TCP connections over shared links and have compared our analytical results against those obtained through simulations. We find that as long as the network is operated without congestion, our analytical approach can predict system performance with reasonable accuracy.

Chapter 1

Introduction

Modeling and performance studies are major issues in design and implementation of networked communication systems and the protocols associated with it. There are many ways to model and study the performance of such systems and protocols. Queuing theory may be used to provide the basic framework and mathematical tools for this purpose. TCP/IP is a protocol suit that provides the rule to transfer a data entity from one computer to another computer throughout the world. This protocol hides the technological differences between networks and makes interconnection independent of underlying hardware. TCP/IP is a network level protocol that provides foundation for peer to peer communication. Its ability to glue hetrogeneous local and wide area networks together makes it a capable integrator. There are two methods of application-to-application interaction - i.e. connection-oriented and connectionless. **Connection-oriented** communication is appropriate when the applications need a sustained interchange of stream of data. In contrast, applications engaged in **connectionless** communications exchange standalone message. The TCP in TCP/IP stands for transmission control protocol that provides reliable, connection-oriented, peer-to-peer communications. TCP allows multiple application programs on a given machine to communicate concurrently, and it demultiplexes incoming TCP traffic among application programs. TCP uses *protocol port* numbers to identify the ultimate destination within a machine. Each port is assigned a small integer used to identify it. TCP uses the connection, not the protocol port as its fundamental abstraction. Connections are identified by a pair of endpoints. TCP defines endpoints to be a pair of integers (*host, port*), where *host* is the IP address for a host and *port* is a TCP port on that host. Terminal login sessions and file transfers are carried over TCP We may view TCP as providing data calls analogous to voice telephone calls. A caller identifies the destination. At the other end, a listening application is alerted that there is an incoming call and picks up the connection. The two ends exchange information for a while. When they are finished, both parties say "goodbye" and hang up. TCP provides the flow control that enables the receiver to regulate the rate at which the sender may transmit data To make the service reliable TCP uses the sliding window principle [2] TCP also contain mechanisms that let it respond to network conditions, adjusting its own behavior to optimize performance. The bandwidth and delay of the underlying network impose limits on throughput.

In this thesis, we present a queueing network model for analyzing the performance of multiple TCP connections. In a typical TCP connection, a data entity may have to traverse several nodes before it reaches its destination. Each TCP connection represents a virtual route consisting of all the nodes between source to destination The connection between two nodes, called link is modeled as a queue. The window flow mechanism of each TCP connection is taken care by semaphore queues as described in [1] Semaphore permits us to model node-to-node and end-to-end window flow controls. The flow and congestion control is taken by window flow control mechanism for each end to end connection. Multiple TCP connections have multiple window flow control. The problem of analyzing multiple window flow control mechanisms can be formulated as a multiple class queueing network with a population constraint for each class. In each class the total number of packets allowed in a subnetwork cannot exceed a predefined number called window size of that class. A subnetwork consists of all the nodes that make a particular TCP connection. A link may be shared by two or more than two TCP connections. In the model these links have traffic of all the classes. In Chapter 2 we introduce the concept of semaphore queue that is the basis of our models. An approximate solution to a queueing network involving one semaphore queue and multiple semaphore queues is also discussed in this. Chapter 3 describes the methods used for performance analysis of TCP connection We have described our proposed algorithm in this chapter. A computed result of the various examples and their validation against simulated results is also given in this. Finally the summary and the suggestions for the future work are given in Chapter 4.

Chapter 2

Semaphore Queues

2.1 Introduction

Semaphore queues are basically used for modeling the window flow control mechanisms [1] A semaphore queue can be considered as a means of controlling the number of customers in a queueing network. We can use it in modeling of a computer communication system consisting of many virtual routes with a sliding window. Associated with this type of system there is a pool of W tokens where W is the window size A customer cannot traverse the network unless it has a token. Customers arriving at the network first see the token queue. If there is no token available in the token queue, the customer is queued in the input queue. Otherwise if tokens are available in the token queue, the customers have the permission to traverse the network Upon departure of a customer from the network, the token is returned back to the pool, after some delay called acknowledgment delay. This queueing system is analyzed approximately by reducing the network to a flow equivalent server. The window flow control mechanism has been modeled as an open queueing system controlled by pool of tokens, as described in [3].

In this chapter, we present an overview of semaphore queues that is the base of our modeling for transmission control protocol (TCP). In Sec. 2 2 we introduced the concept of semaphore queues. An approximate solution to a queueing network involving one semaphore queue is given in Sec 2.3. In Sec 2.4 we give an approximate algorithm for analyzing a queueing network with multiple semaphore queue.

2.2 A Queueing Network With Single Semaphore

A semaphore station consists of an input queue f(S) and a token queue e(S). Customer arriving at the semaphore queue requests for a token. The customer departs immediately, if there is a token available in token queue e(S). Otherwise, the customer is blocked and it is forced to wait in the input queue f(S) until a token becomes available.

Therefore if there are tokens in token queue e(S), then there are no customers in the input queue On the other hand, if there are customers in the input queue, then e(S) is empty.

A customer having received a token leaves the input queue and enters network 1 as shown in Fig 2 1 The total number of tokens available is fixed to C i e. network 1 can be used at most by C customers. Also at any time, the customers in network 1 plus the returning tokens in network 2 will be less than or equal to C.

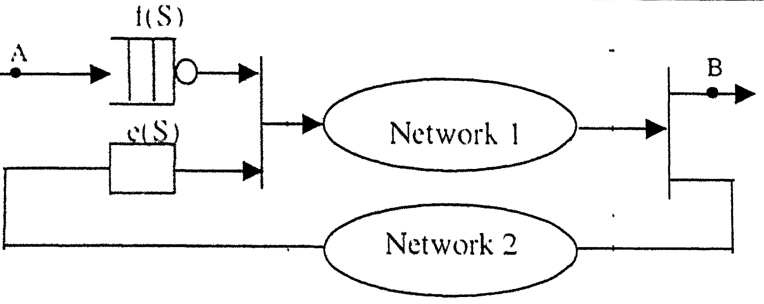

Fig 2.1 A queueing network with a single semaphore

Networks 1 and 2 are assumed to be of BCMP (Basket, Chandy, Muntz, and Palacios) type [7]. Customers arrive at the semaphore queue in a Poisson fashion at rate λ.

At the instant the queue f(S) and e(S) contain a customer each, the two customers instantaneously depart from their respective queues and merge into a single operation. This operation can be fully depicted by the join symbol as shown in Fig 2.2 of Fork-Join queue [4].

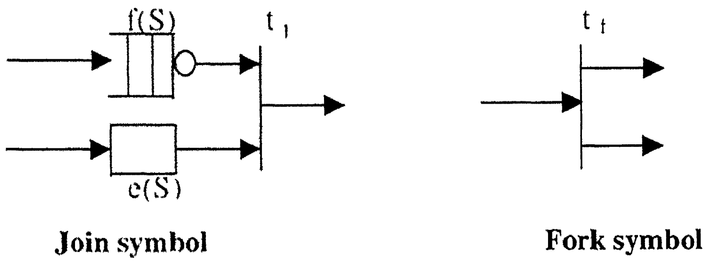

Fig 2.2 Fork - join symbols

Similarly Join symbol as shown in Fig 2.2 depicts that when a customer arrives at the point when customer departs from network 1 is spilt into two siblings. Fork - Join queues deals with the situations when a given job is decomposed and distributed to be serviced in parallel and are aggregated after the service completion of decomposed jobs. Fork process corresponds to the decomposition and service in parallel and the Join process corresponds to the aggregation after completion of all the subjobs.

2.3 Solution of the Semaphore Queue

Consider the queueing network described in Fig 2 1. An exact analysis of this model is rather difficult. It can be analyzed approximately using decomposition and aggregation technique To analyze the single semaphore queueing network, we first explain the system shown in Fig 2.3 assuming that arrival process at queue e(S) is described by a state dependent arrival rate γ(k) where k is the number of outstanding tokens

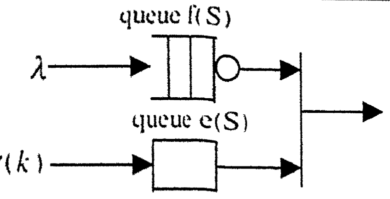

Fig 2.3 The semaphore queue

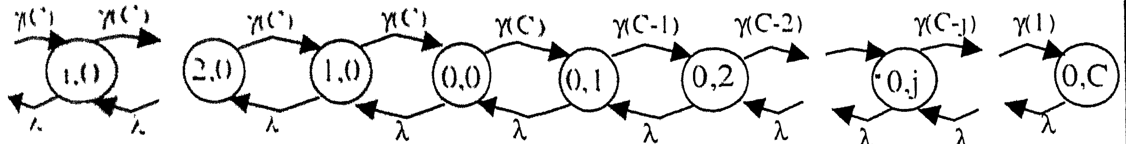

Fig 2.4 Rate diagram of semaphore queue

The state of the system shown in Fig 2.3 can be described by tuple (i, j) where i is the number of customers in queue f (S) and j is the number of tokens in queue e(S). The rate diagram associated with this system is shown in Fig 2.4. The rate diagram of this system is identical to the rate diagram of M / M / 1 queue [5] with arrival rate λ and a state dependent service rate γ(n) if $n \leq C$, and γ(C) if $n > C$. Where n is number of customers in this M / M / 1 queue and C is the total number of tokens available.

The solution of this system is obtained by applying the classical results of M / M / 1 queue. Thus, we have

$$p(i,0) = \rho^i \; p(0,0), \tag{2.1}$$

$$p(0,j) = \frac{\Pi(j)}{\lambda^j} \; p(0,0); \tag{2.2}$$

where $\rho = \lambda / \gamma(C)$ and

$$\Pi(j) = \begin{cases} \prod_{k=0}^{j-1} \gamma(C-k) & j > 0 \\ 1 & j = 0 \end{cases}$$

The probability $p(0,0)$ is obtained by Conversion relation [4].

By conversion relation,

$$\sum_{i=0} \sum_{j=0}^{C} p(i,j) = 1 \tag{2.3}$$

$$\Rightarrow \quad \sum_{i=0} p(i,0) + \sum_{j=1}^{C} p(0,j) = 1 \tag{2.4}$$

$$\Rightarrow \quad p(0,0) \sum_{i=0}^{\infty} \rho^i + p(0,0) \sum_{j=1}^{C} \frac{\Pi(j)}{\lambda^j} = 1$$

$$\Rightarrow \quad p(0,0)^{-1} = \frac{1}{1-\rho} + \sum_{j=1}^{C} \frac{\Pi(j)}{\lambda^j} \tag{2.5}$$

We can obtain the marginal probabilities for each queue Index 1 is for queue f(S) and 2 is for queue e(S).

$$p_1(0) = \sum_{j=0}^{C} p(0,j)$$

$$= p(0,0) + \sum_{j=1}^{C} p(0,j)$$

$$= p(0,0) + p(0,0) \sum_{j=1}^{C} \frac{\Pi(j)}{\lambda^j} \tag{2.6}$$

By putting the value of $p(0,0)$ from equation 2.5 to equation 2.6 we get,

$$p_1(0) = \frac{1-\rho(1+p(0,0))}{1-\rho} \tag{2.7}$$

$$p_1(i) = \rho^i p(0,0) \qquad i > 0 \tag{2.8}$$

$$p_2(0) = \frac{1}{1-\rho} p(0,0) \tag{2.9}$$

$$p_2(j) = \frac{\Pi(j)}{\lambda'} p(0,0) \quad 0 \le j \le C \tag{2 10}$$

Now, expression (2.1) to (2.10) are obtained assuming that $\gamma(k)$ is known. This can be approximately obtained by studying the closed queueing network call it Q obtained by linking queueing networks 1 and 2 as shown in Fig 2.4.

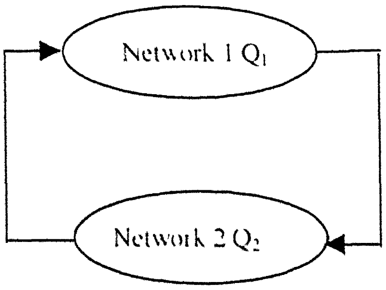

Fig 2.5 A closed queueing network Q

The analysis of the queueing network can be carried out easily seeing that we have assumed that network 1 and 2 are of the BCMP type. Therefore we can calculate the throughput of queueing network Q with k customers where k=1,2,3.. C. Then the throughput of this closed queuing network Q will give the arrival rate $\gamma(k)$ of the tokens at the token queue e(S). Now the mean number of customer in closed queueing network can be obtained as follows.

2.3.1 Mean Number of Customer in Closed Queueing Network Q

Let $p_1(m/h)$ be the probability that there are m customers in Q_1 i.e. in network 1 given that there are h customers in the closed queueing network Q, $p_1(m)$ be the probability that there are m customers in Q_1 and $q(j)$ is the marginal probability distribution that there are j customers in token queue e(S).

For $m > h$,

$$p_1(m/h) = 0$$

$$\Rightarrow \quad p_1(m) = 0\ .$$

For $0 < m \le h$,

$$p_1(m) = \sum_{h=m}^{C} p_1(m/h)q(C-h) \qquad m = 1,2\ldots\ldots C \qquad (2.11)$$

Hence, the mean number of customers in Q_1 is

$$L_{Q_1} = \sum_{m=1}^{C} mp_1(m)$$

$$= \sum_{m=1}^{C} m\left(\sum_{h=m}^{C} p_1(m/h)q(C-h)\right)$$

$$= \sum_{h=1}^{C} q(C-h)\sum_{m=1}^{C} mp_1(m/h) \qquad (2.12)$$

The quantity $\sum mp_1(m/h)$, summed over $m = 1,2\ .C$, is the mean number of customer in Q_1 given there are h customers in Q. Now if the mean response time of Q_1 as a function of the number of customers h in Q is $R(h)$. Thus,

$$\sum_{m=1}^{C} mp_1(m/h) = \gamma(h)R(h)$$

Where $\gamma(h)$ is the rate at which tokens return to token queue c(S).

Hence from expression (2.12)

$$L_{Q_1} = \sum_{h=1}^{C} q(C-h)\gamma(h)R(h) \qquad (2.13)$$

2.3.2 Condition of Solution Existence

The solution existence condition can be expressed as,

$$\lambda < \gamma(C) \qquad (2.14)$$

Where $\gamma(C)$ is the maximum throughput of the network Q. This condition is the same as stability condition derived from Lavenberg's theorem [6].

2.4 The Approximate Algorithm for Solving A Queueing Network With Single Semaphore:

Assumptions:

- Input queue has infinite buffer capacity
- The service discipline in the Input queue f(S) is FCFS
- Network 1 and Network 2 are assumed to be of BCMP type
- Customer arrives at queue f(S) in a Poisson fashion at rate λ .
- Arrival process at queue e(S) is described by state-dependent arrival rate $\Upsilon(k)$
- Interarrival times at queue e(S) are exponentially distributed with a rate $\Upsilon(k)$ where k is number of outstanding tokens.

Symbols Used :

C - total number of tokens or permissions available.

k - number of outstanding tokens i.e the number of tokens in queue e(S).

λ - Poisson arrival rate of customers at the input queue.

$\gamma(k)$ - state dependent arrival rate at the token queue e(S).

Step0: Evaluate the marginal probabilities of the input queue and token queue by using equation (2.7) to (2.10).

Step1. Obtain the state dependent arrival rate $\gamma(k)$ by solving the closed queueing network Q shown in Fig 2 4 where k varies from 1,2...C.

Step3: Compute the mean number of customer $L_{f(S)}$ in input queue e(S) by using the relation,

$$L_{f(S)} = \sum_{i=1}^{\infty} ip(i) \qquad (2.15)$$

Where $p(i)$ is the marginal probability distribution if there are i customers in queue f(S)

Step4· Find out the mean number of customer L_{Q_1} in queuing network Q1 by using the equation 2.13

Step5 Finally compute the mean response time T for a customer traversing from A to B in Fig 2 1 by using Little's formula [5]

$$T = \frac{1}{\lambda}\left[L_{f(S)} + L_{Q_1}\right] \tag{2 16}$$

2.5 Analysis of Queueing Network With Multiple Semaphores

Generally semaphore queue can be considered as a means of controlling the number of customers in a queueing network. These queueing networks can be combined by embedding one network within another to make a larger more complex system. In this section we give a method for computing the solution of such multiple semaphore queueing networks. For example we consider the nested queueing network shown in Fig 2.6. In this figure $QN_{i,n}, i = 1,2,3,4$ are four arbitrary BCMP queueing networks, and $S_{i,n-1}, i = 1,2$ are two semaphore controlled queueing networks as shown in Fig 2.7. The index n refers to a level of semaphore control.

Let C_n and C_{n-1} be the total number of tokens associated with the nth and (n-1) th level semaphore controlled queueing networks respectively. Semaphore controlled queueing networks $S_{i,n-1}, i = 1,2$ themselves may contain other lower levels of semaphore queues. Let us first consider the lowest level semaphore controlled queueing network $S_{1,n-1}$ as shown in Fig 2 7 As in section 2.3 we can link networks $Q_{1,n-1}$ and $Q_{2,n-1}$ to form a closed BCMP queueing network, call it Q_{n-1} as shown in Fig 2.8.

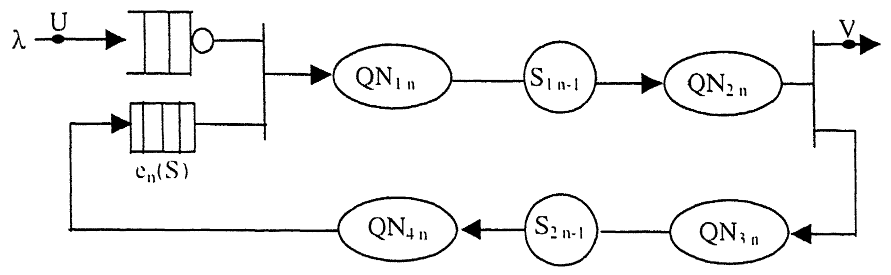

Fig 2.6 A level n semaphore controlled queueing network

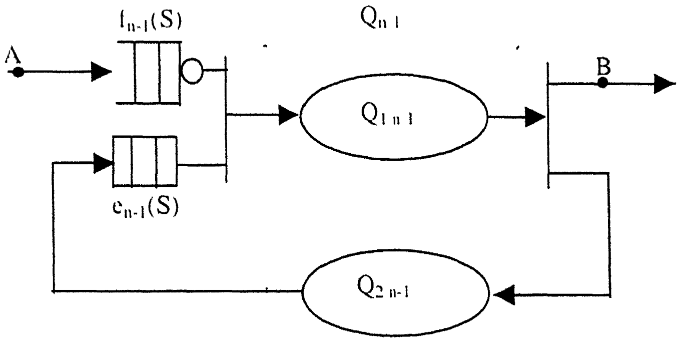

Fig 2.7 A level (n-1) semaphore subnetwork

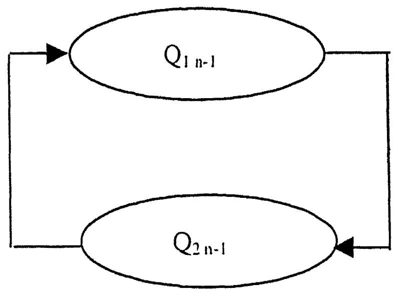

Fig 2.8 A closed queueing network Q_{n-1}

This closed queueing network can be analyzed using the mean value analysis (MVA) algorithm in order to obtain $R_{n-1}(k)$ and $\gamma_{n-1}(k)$ Here $R_{n-1}(k)$ is the mean time to traverse network $Q_{1,n-1}$ as a function of number of customer k in Q_{n-1} and $\gamma_{n-1}(k)$ is the mean throughput of closed queueing network Q_{n-1} as a function of k. Mean throughput $\gamma_{n-1}(k)$ can also be considered as the rate at which tokens return back to the token queue $e_{n-1}(S)$ where k is the number of outstanding tokens. Obviously k varies from 1 to C_{n-1}. Hence the mean response time $R_{n-1}(c)$ of a customer between points A and B conditioned that he finds c customers (including himself) in queue $f_{n-1}(S)$ and in Q_{n-1} upon arrival is approximately given by,

$$R_{n-1}(c) = \begin{cases} R_{n-1}(c) & \\ R_{n-1}(C_{n-1}) + (c - C_{n-1})/\gamma_{n-1}(C_{n-1}) & c \geq C_{n-1} \end{cases} \qquad (2.17).$$

The above expression can be explained as fallows. If $c \leq C_{n-1}$ then all the customers are in Q_{n-1}. Thus our customer is delayed by $R_{n-1}(c)$. If $c > C_{n-1}$ then only C_{n-1} customers are in Q_{n-1} and remaining ($c - C_{n-1}$) are waiting in queue $f_{n-1}(S)$. So this can be seen as receiving a mean service time equal to $1/\gamma_{n-1}(C_{n-1})$ before it enters $Q_{1,n-1}$ where it is delayed on the average by $R_{n-1}(C_{n-1})$. Thus we can obtain the above expression for $R_{n-1}(c)$ when $c \geq C_n$.

Now in Fig 2.6 we can approximately substitute $S_{1,n-1}$ by a flow equivalent infinite server queue with a state dependent mean service time equal to $R_{n-1}(c)$ where $c = 0,1....C_n$. Following similar logic we can also approximately substitute $S_{2,n-1}$ by another flow equivalent infinite server queue. Now $Q_{1,n}$ and $Q_{2,n}$ are the queueing networks consisting of $QN_{1,n}$, $S_{1,n-1}$, $QN_{2,n}$ and $QN_{3,n}$, $S_{2,n-1}$, $QN_{4,n}$ respectively. Then $Q_{1,n}$, $Q_{2,n}$ and the closed queueing network consisting of $Q_{1,n}$, and $Q_{2,n}$ are all BCMP type queueing networks. Thus we can analyze it in the same way as a queueing network with single semaphore as described in Sec 2.4

Chapter 3

Performance Evaluation of Multiple TCP Circuits

2.1 Introduction

We use transmission control protocol (TCP) when two application programs transfer a large volume of data from one machine to another. In these TCP connections data is defined as a stream of bits, divided into octets called bytes. To provide reliable stream delivery service guarantee TCP uses a technique called sliding window protocols. In normal communication technique, when receiver receives any packet it sends back an acknowledgement (ACK) message for that particular packet. The sender keeps record of each packet it sends and waits for an ACK before sending the next packet. On the other hand sliding window protocol places a small, fixed size predefined *window* on the packet sequence and transmits all packets that lie inside the window. Thus this protocol allow the sender to transmit multiple packets before waiting for an acknowledgement (ACK) Whenever first packet reaches to its destination it sends back an ACK to the sender Once the sender receives an acknowledgement for the first packet inside the window, it slides the window along and sends the next packet. The window continues to slide as long as acknowledgements (ACKS) are received. In case new acknowledgement stops coming then transmitter times out and sends all the packets lying within the window.

In this chapter we have presented our modeling scheme and method for computing the performance analysis of multiple TCP connections. In Sec 3.2 we have introduced the concept of multiple TCP connection. In Sec 3.3 we have explained our modeling scheme. Method used for solving two Semaphore queues with two class queueing network is given in Sec. 3.4 We have also discussed the Mean Value Analysis algorithm for solving the multiclass closed network in Sec 3.5. Finally, In Sec. 3.6 we have presented a general approximate algorithm for evaluating the performance of N TCP connections.

3.2 Multiple TCP Connections

Transmission control protocol (TCP) allows multiple application programs on a given machine to communicate concurrently and it demultiplexes incoming TCP traffic among application programs. TCP uses *protocol port* numbers to identify the ultimate destination within a machine. Each port is assigned a small integer used to identify it. A pair of endpoints identifies a particular TCP connection. TCP defines *endpoints* to be a pair of integers (*host, port*) where *host* is the Internet address for the host and *port* is a TCP port on that host. For the example the endpoint (144.16.167.5, 10) specifies TCP port 10 on the machine with IP address 144.16.167.5. A connection is defined by its two endpoints. Thus if there is a TCP connection from machine (144.16.167.100) to machine (144 16 167.6) it might be defined by the endpoints:

(144.16.167.100, 50) and (144.16.167 6, 90)

Because TCP identifies a connection by a pair of endpoints a given TCP port number can be shared by multiple connection on the same machine. For example we can add another connection to the machine (144.16.167.6) from machine (144 16.167.200):

(144 16 167.200, 60) and (144 16.167.6, 90).

Thus an endpoint might be shared in the multiple TCP connections

3.3 Modeling of Multiple TCP Connections with Shared Link

As we have seen in the above section that a link might be shared in multiple TCP connections We can approximately model the multiple TCP connections of partially shared link with a sliding window protocol by using semaphore queues. Semaphore permits us to model end to end window flow controls. Each TCP connection represents a virtual route consisting of all the nodes between source to destination. The flow and congestion control is taken care by window flow control mechanism for each end to end connection. The links are modeled as a infinite buffer queue. Multiple TCP connections have multiple window flow control. The multiple window flow control can be modeled as multiple class queueing networks with a population constraint for each class. In each class the total number of packets allowed in a subnetwork cannot exceed a predefined number called *window size* of that class. A subnetwork is a queueing network made up by the links that makes a particular TCP connection where a link is replaced by an infinite buffer queue Whenever any link is shared by the multiple classes, it has the traffic of all the classes. A particular class semaphore station consists of an input queue and a token queue. Every class has its own window size that is equal to the number of tokens available for that class or connection.

Let us take an example to explain modeling concepts in a broader way. The multiple TCP connection under study consists of three nodes with two TCP connections as shown in Fig. 3.1

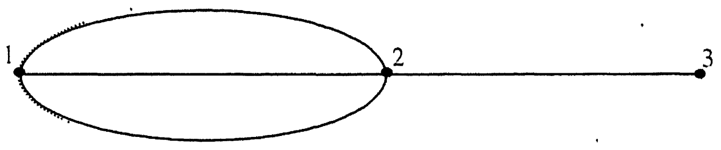

Fig. 3.1 An example of two TCP connections

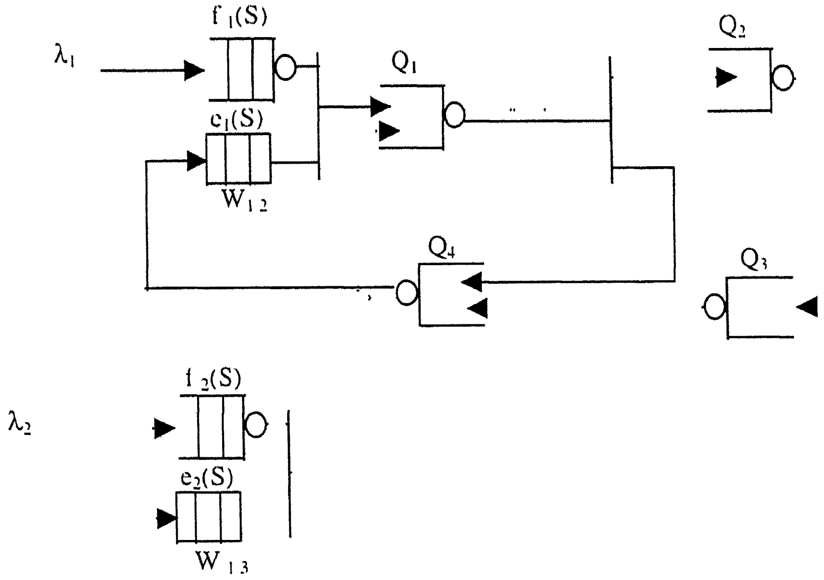

Fig 3.2 Queueing model of the TCP connections shown in Fig 3.1

The first TCP connection is from node 1 to node 2 and the second TCP connection is from node 1 to node 3. The link (1-2) is shared by the two TCP connections. For every TCP connection we need one semaphore station to control its end to end window flow control. So we can model the above two TCP connections with two semaphore stations having two class of traffic. Class 1 traffic represents the traffic associated with first TCP connection and class 2 traffic denotes the traffic associated with second TCP connection. Each TCP connection has its own window size and connections are of piggyback type. In these type of connections receiver sends acknowledgement (ACK) via same virtual route through which packets are coming. The approximate model of the above connection is shown in Fig 3 2.

In the figure class 1 traffic is represented by continuous lines (——) and class 2 traffic is shown by breaken lines ( ). Each class semaphore station (S) consists of input queue f(S) and a token queue e(S). The link between two nodes is replaced by an infinite buffer queue. The number of tokens initially available for any particular class is equal to the window size of that particular class or connection. W_{12} is the number of tokens available (window size) for the TCP connection from node 1 to node 2 and W_{13} denotes the number of tokens available for the TCP connection from node 1 to node 3. Whenever packet of any class reaches at its destination the tokens associated with that class is returned back via the network of reverse queues. Now the above queueing model have two semaphore stations with two class of queueing network. We can compute the performance parameter of these TCP connections by solving it. Performance parameter includes the Mean number of packets in the system and Mean delay experienced by the packet in the system.

3.4 Methods Used for Solving Two Semaphore Queues with Two Class Queueing Network

Consider the queueing network described above and shown in Fig 3 2. The network consists of two class of Semaphore station The class 1 semaphore station consists of an input queue $f_1(S)$ and token queue $e_2(S)$. In the same way class 2 semaphore station also have an input queue $f_2(S)$ and token queue $e_2(S)$. When a packet of any particular class (class 1 or class 2) arrives at the semaphore queues, it requests for a token of that particular class. If a token is available of the associated class, the packet immediately departs with that token. If there is no token available in the token queue of corresponding class, the packet is blocked and forced to wait in the input queue of that particular class. This semaphore operation is described in mathematical form by two semaphore queues of different class shown in Fig3.3.

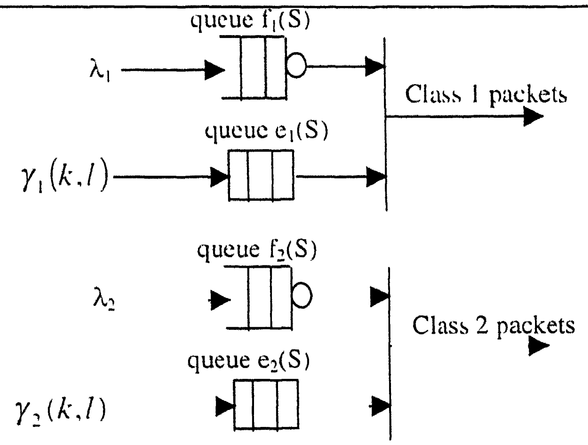

Fig 3.3 Two Semaphore Stations with Two Class

In particular, we first analyze the system shown in Fig 3.3 assuming that the arrival processes at queue $e_1(S)$ and $e_2(S)$ is described by state dependent arrival rates $\gamma_1(k,l)$and $\gamma_2(k,l)$respectively. Where k and l are the outstanding tokens for class 1 and class 2 respectively. The system depicts the semaphore operation described above. The arrival process at queue $f_1(S)$ and $f_2(S)$ is assumed to be Poisson distributed. There are C_1 number of class 1 tokens and C_2 number of class 2 tokens. We also assume that the interarrival times at queues $e_1(S)$ and $e_2(S)$ are exponentially distributed

The state of the system in equilibrium can be described by tuple (i,j,m,n), where i is the number of class 1 packets in queue $f_1(S)$, j is the number of class 1 tokens in queue $e_1(S)$, m is the number of class 2 packets in queue $f_2(S)$ and n is the number of class 2 tokens in queue $e_2(S)$. We have assumed that the input queue of each class

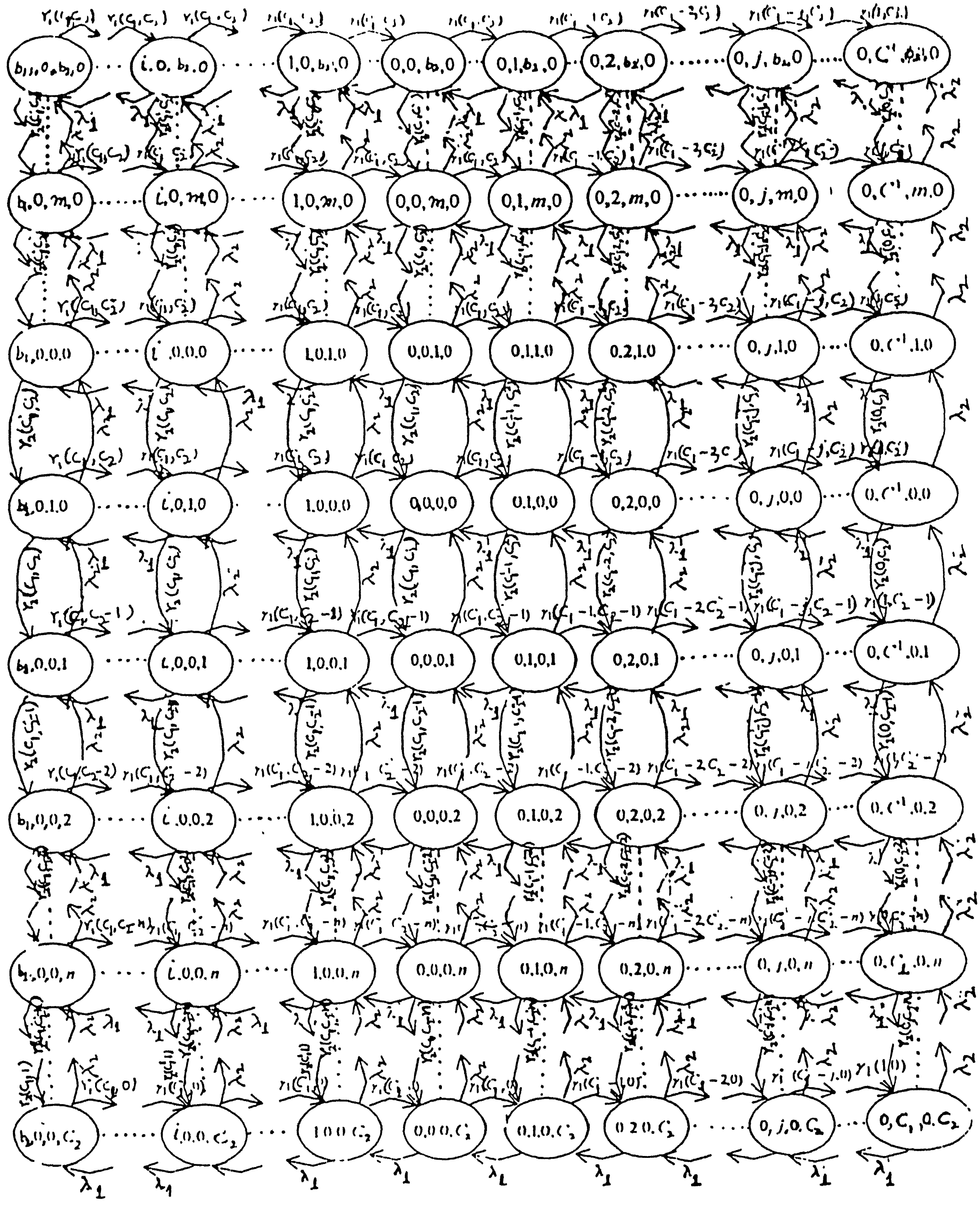

Fig 3·4 Rate diagram of queueing system shown in Fig 3.2

semaphore queues is a finite capacity. The buffer capacity of class 1 input queue is b_1 and the buffer capacity of class2 input queue is b_2. The rate diagram associated with this system is shown in Fig 3 4. From this rate diagram we can write the state equations as fallow,

$$p(0,C_1,0,C_2)(\lambda_1+\lambda_2)-\gamma_1(1,0)p(0,C_1-1,0,C_2)-\gamma_2(0,1)p(0,C_1,0,C_2-1)=0 \qquad (3.1)$$

$$p(0,j,0,C_2)[\gamma_1(C_1-j,0)+\lambda_1+\lambda_2]-\lambda_1 p(0,j+1,0,C_2)-\gamma_1(C_1-j+1,0)p(0,j-1,0,C_2)$$
$$-\gamma_2(C_1-j,1)p(0,j,0,C_2-1)=0 \qquad j=C_1-1,C_1-2,\dots 2,1 \qquad (3.2)$$

$$p(i,0,0,C_2)[\lambda_1+\lambda_2+\gamma_1(C_1,0)]-\lambda_1 p(i-1,0,0,C_2)-\gamma_1(C_1,0)p(i+1,0,0,C_2)$$
$$\gamma_2(C_1,1)p(i,0,0,C_2-1)=0 \qquad i=0,1,2\dots(b_1-1) \qquad (3.3)$$

$$p(b_1,0,0,C^2)[\lambda_2+\gamma_1(C_1,0)]-\lambda_1 p(b_1-1,0,0,C_2)-\gamma_2(C_1,1)p(b_1,0,0,C_2-1)=0 \qquad (3.4)$$

$$p(0,C_1,0,n)[\gamma_2(0,C_2-n)+\lambda_1+\lambda_2]-\gamma_1(1,C_2-n)p(0,C_1-1,0,n)-\lambda_2 p(0,C_1,0,n+1)$$
$$-\gamma_2(0,C_2-n+1)p(0,C_1,0,n-1)=0 \qquad n=C_2-1,C_2-2,\dots 2,1 \qquad (3.5)$$

$$p(0,j,0,n)[\lambda_1+\lambda_2+\gamma_1(C_1-j,C_2-n)+\gamma_2(C_1-j,C_2-n)]-\lambda_1 p(0,j+1,0,n)-\lambda_2 p(0,j,0,n+1)$$
$$-\gamma_2(C_1-j,C_2-n+1)p(0,j,0,n-1)-\gamma_2(C_1-j+1,C_2-n)p(0,j-1,0,n)=0 \qquad (3.6)$$

$$\text{where}\begin{cases} j=C_1-1,\ \ 2,1 \\ n=C_2-1,\ .\ 2,1 \end{cases}$$

$$p(i,0,0,n)[\lambda_1+\lambda_2+\gamma_1(C_1,C_2-n)+\gamma_2(C_1,C_2-n)]-\lambda_1 p(i-1,0,0,n)-\gamma_2(C_1,C_2-n+1)p(i,0,0,n-1$$
$$-\gamma_1(C_1,C_2-n)p(i+1,0,0,n)=0 \qquad (3.7)$$

$$\text{where}\begin{cases} i=1,2\ ..(b_1-1) \\ n=C_2-1,\ \ .1 \end{cases}$$

$$p(b_1-1,0,0,n)[\lambda_2+\gamma_1(C_1,C_2-n)+\gamma_2(C_1,C_2-n)]-\lambda_2 p(b_1-1,0,0,n+1)-\lambda^1 p(b_1-1,0,0,n)$$

$$-\gamma_2(C_1,C_2-n+1)p(b_1-1,0,0,n-1)=0 \qquad n=C_2-1,C_2-2,\dots,2,1 \qquad (3.8)$$

$$p(0,C_1,m,0)[\lambda_2+\lambda_1+\gamma_2(0,C_2)]-\lambda_2 p(0,C_1,m-1,0)-\gamma_1(1,C_2)p(0,C_1-1,m,0)$$

$$-\gamma_2(0,C_2)p(0,C_1,m+1,0)=0 \qquad m=0,1,2\dots(b_2-1) \qquad (3.9)$$

$$p(0,j,m,0)[\lambda_1+\lambda_2+\gamma_1(C_1-j,C_2)+\gamma_2(C_1-j,C_2)]-\lambda_1 p(0,j+1,m,0)-\lambda_2 p(0,j,m-1,0)$$
$$-\gamma_1(C_1-j+1,C_2)p(0,j-1,m,0)-\gamma_2(C_1-j,C_2)p(0,j,m+1,0)=0 \qquad (3.10)$$

Where $\begin{cases} j = C_1 - 1, C_1 - 2, \ldots 2, 1 \\ m = 0, 1, \ldots, (b_2 - 1) \end{cases}$

$$p(i,0,m,0)[\lambda_1 + \lambda_2 + \gamma_1(C_1,C_2) + \gamma_2(C_1,C_2)] - \lambda_1 p(i-1,0,m,0) - \lambda_2 p(i,0,m-1,0)$$

$$-\gamma_1(C_1,C_2)p(i,0,m+1,0) - \gamma_2(C_1,C_2)p(i,0,m+1,0) = 0 \qquad (3.11)$$

where $\begin{cases} i = 0, 1 \ldots, (b_1 - 1) \\ m = 0, 1 \ldots, (b_2 - 1) \end{cases}$

$$p(b_1,0,m,0)[\lambda_2 + \gamma_1(C_1,C_2) + \gamma_2(C_1,C_2)] - \lambda_1 p(b_1 - 1,0,m,0) - \lambda_2 p(b_1,0,m-1,0)$$

$$-\gamma_2(C_1,C_2)p(b_1,0,m+1,0) = 0 \qquad m = 0,1,\ldots,(b_2 - 1) \qquad (3.12)$$

$$p(0,C_1,b_2,0)[\lambda_1 + \gamma_2(0,C_2)] - \lambda_2 p(0,C_1,19,0) - \gamma_1(1,C_2)p(0,C_1 - 1,b_2,0) = 0 \qquad (3.13)$$

$$p(0,j,b_2 - 1,0)[\lambda_1 + \gamma_2(C_1 - j,C_2) + \gamma_1(C_1 - j,C_2)] - \lambda_1 p(0,j+1,b_2,0) - \lambda^2 p(0,j,b_2 - 1,0)$$

$$-\gamma_1(C_1 - j + 1,C_2)p(0,j-1,b_2,0) = 0 \qquad j = C_1 - 1, \ 2,1. \qquad (3.14)$$

$$p(i,0,b_2,0)[\lambda_1 + \gamma_1(C_1,C_2) + \gamma_2(C_1,C_2)] - \lambda_1 p(i-1,0,b_2,0) - \lambda_2 p(i,0,b_2 - 1,0)$$

$$-\gamma_1(C_1,C_2)p(i+1,0,b_2,0) = 0 \qquad i = 0,1,2,\ldots,(b_1 - 1) \qquad (3.15)$$

$$\sum_{i=0}^{b_1}\sum_{j=1}^{C_1}\sum_{m=0}^{b_2}\sum_{n=1}^{C_2} p(i,j,m,n) = 1 \qquad (3.16)$$

After solving the simultaneous equations (3.1) to (3.16) we get the probability of each state shown in Fig 3.4. Now we can easily calculate the marginal probability for the input queue and token queue of each class by following relations.

$$p_{I_1(S)}(i) = \sum_{j=0}^{C_1}\sum_{m=0}^{b_2}\sum_{n=0}^{C_2} p(i,j,m,n) \qquad (3.17)$$

$$p_{I_2(S)}(m) = \sum_{i=0}^{b_1}\sum_{j=0}^{C_1}\sum_{n=0}^{C_2} p(i,j,m,n) \qquad (3.18)$$

$$p_{c_1(S)}(j) = \sum_{i=0}^{b_1}\sum_{m=0}^{b_2}\sum_{n=0}^{C_2} p(i,j,m,n) \qquad (3.19)$$

$$p_{c_2(S)}(n) = \sum_{i=0}^{b_1}\sum_{j=0}^{C_1}\sum_{m=0}^{b_2} p(i,j,m,n) \qquad (3.20)$$

Where $p_{I_1(S)}(i), p_{I_2(S)}(m), p_{c_1(S)}(j), p_{c_2(S)}(n)$ are the probability that there are *i* packets in the input queue of class 1, probability that there are *m* packets in the input queue of class

2, probability that there are j number of tokens in the token queue of class 1 and probability that there are n number of tokens in token queue of class 2 respectively.

Now, expression (3.1) to (3.16) are obtained assuming that $\gamma_1(k,l)$ and $\gamma_2(k,l)$ are known, where k and l are the number of outstanding tokens for class 1 and class 2 respectively. This can be approximately obtained by solving the two class closed queueing network (call it Q) obtained by linking the queueing networks of class 1 and class 2 as shown in Fig 3.5. The dark line represents class 1 and the dash line represents class2.

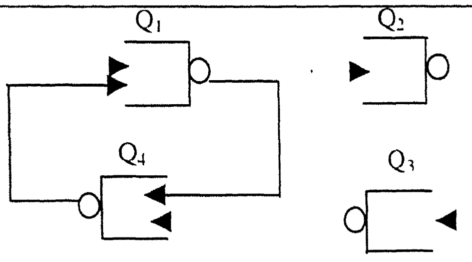

Fig 3.5 A two class closed queueing network Q for the TCP connections shown in Fig 3.2

We can calculate the throughputs of class 1 and class 2 queueing networks with k and l packets of class 1 and class 2 respectively, where $k = 0,1.......C^1$ and $l = 0,1.. ..C^2$. Then class 1 throughput of this network Q will give the arrival rate $\gamma_1(k,l)$ and class 2 throughput will give the arrival rate $\gamma_2(k,l)$. We can solve this multiclass closed queueing networks by using Mean Value Analysis (MVA) technique as given below.

3.5 Mean Value Analysis (MVA) Algorithm for solving the multiclass closed queueing networks

In the multiple class closed network all the jobs of all different classes are to be circulated inside the system. There will be no external arrivals to the system and hence no departures out of the system. The technique used to evaluate the closed queueing network is the mean value analysis (MVA) technique. We can calculate the parameter mean waiting time and mean throughput at each queue in closed multiple class queuing networks.

The MVA algorithm is based on the Mean Value Theorem [7]. This theorem states that a customer of class k arriving to a queue in the network would see the same average number in the queue as an external observer would see if the network had one less customer of class k The concept of visit ratio is used in the algorithm. Visit ratio at node k, V_k is the ratio of throughput of node k to the throughput of the reference node in the network For the analysis of closed networks node 1 is arbitrary selected as reference node In mean value analysis, we first initialize and compute the waiting time at each queue for each class in the network. Next we compute the throughput of each class, followed by mean queue length at each queue for each class in the network. Finally we update the queue length distribution. These parameters are computed iteratively until we reach the total population of each class

Assumptions:

- All the queues in the network have infinite buffer capacity.
- There are no external arrivals to any of the queues for any of the classes.
- Classes are independent and the routing probabilities for each class in the network are static.
- Exponential service times for all classes at all nodes in the network.
- There is FCFS type of service discipline in the network
- There is at least one class of jobs with a non-zero population.

Symbols Used:

N- number of nodes in the network

R- total number of classes

m - number of servers at node i for i=1,2,...,N

τ_{ik} - mean service time at node of class k for $i = 1,2,.......N$ & $k = 1,2,.....,R$

$E[N_{ik}]$ - mean number of class k customers at node i for $i = 1,2,...,N$ & $k = 1,2...R$.

$E[W_{ik}]$ - mean waiting time of class k customers at node i for $i = 1,2,\ldots,N$ $\& \; k = 1,2,\ldots,R$

λ_{ik} - mean throughput of class k customers at node i for $i = 1,2,\ldots,N$ $\& \; k = 1,2,\ldots,R$

V_{ik} - visit ratio of class k customers at node i for $i = 1,2 \ldots N \;\&\; k = 1,2,\ldots,R$

$E[N_i] = \sum_{k=1}^{R} E[N_{ik}]$ for $i = 1,2,\ldots,N$

$\underline{C}$ - $\{C_1, C_2, \ldots, C_R\}$, population vector where C_k is total population of class k

$\underline{\mathbf{0}}$ - $\{0,0,\ldots,0\}$, zero vector, zero in each class for all R classes

$\underline{\mathbf{e}_k}$ - $\{0,0,\ldots 1,0,\ldots,0\}$, unit vector in the kth position.

```
Step0: Initialization:
    $LN_i(\underline{\mathbf{0}}) = 1.0$        for $i = 1,2,\ldots,N$
    $P_i(0,\underline{\mathbf{0}}) = 1.0$       for $i = 1,2,\ldots,N$
    $P_i(j,\underline{\mathbf{0}}) = 0.0$       for $i = 1,2,\ldots,N$
Step1. Do steps 3,4 & 5 for $c_1 = 0,1,\ldots,C_1, c_2 = 0,1,\ldots, c_k = 0,1,\ldots,C_k$.
Step2: Do for $k = 1,2,\ldots,R \;\&\; i = 1,2,\ldots,N$
    $W_{ik}(\underline{C}) = \begin{cases} \tau_{ik} V_{ik} LN(\underline{C}) \\ \tau_{ik} V_{ik} \\ \tau_{ik} V_{ik} LN_i(\underline{C}) + \sum_{j=1}^{m_i-1} (m_i - j) P_i(j-1, \underline{C} - \underline{\mathbf{e}_k}) \end{cases}$
    $W_k(\underline{C}) = \sum_{i=1}^{N} w_{ik}(\underline{C})$
    $\lambda_k(\underline{C}) = \frac{c_k}{W_k(\underline{C})}$
Step3: $LN_i(C) = 1 \pm \sum_{k=1}^{R} \lambda_k(C) V_{ik} w_{ik}(C)$  for $i = 1,2,\ldots,N$
Step4: Do for $i = 1,2,\ldots,N \;\&\; j = 1,2,\ldots,m_i - 1$
```

$$P_i(j,\underline{C}) = \left(\frac{1}{j}\right) \cdot \sum_{k=1}^{R} \lambda_k(\underline{C})\, V_{ik}\, \tau_{ik}\, P_i(j-1, \underline{C} - e_k)$$

$$P_i(0,\underline{C}) = 1 - \frac{\sum_{k=1}^{R} \lambda_k(\underline{C})\, V_{ik}\, \tau_{ik} + \sum_{j=1}^{m_i-1} (m_i - j) P_i(j, \underline{C})}{m_i}$$

Step5: Find out the mean number, mean throughput and mean waiting time of each class at each node by following relations

$\underline{C} - \{C_1, C_2, \ldots, C_R\}$

$E[N_i] = LN_i(\underline{C}) - 1$ for $i = 1,2 \ldots, N$

$E[N_{ik}] = \lambda_k(\underline{C})\, w_{ik}(\underline{C})$ for $i = 1,2 \ldots N$ & $k = 1,2,\ldots R$

$E[W_{ik}] = \dfrac{W_{ik}(\underline{C}) - V_{ik}\,\tau_{ik}}{V_{ik}}$ for $i = 1,2,\ldots,N$ & $k = 1,2,\ldots,R$

$\lambda_{ik} = \lambda_k(\underline{C})\, V_{ik}$ for $i = 1,2,\ldots\ldots, N$ & $k = 1,2,\ldots\ldots,R$

3.6 The Approximate Algorithms for Computing the Performance of Multiple TCP Connections

As discussed in Sec 3.3, We can model the two TCP connections by two semaphore stations with two classes of queueing networks. In the same way, we can also model the multiple TCP connections by using multiple semaphore stations with multiple class of queueing networks. A separate class characterizes the traffic of each TCP connection. For every TCP connection there is a fixed virtual route. In the model, each link of a virtual route is replaced by a infinite buffer queue. Thus for N TCP connections we have N class of queueing networks made by the N class of virtual routes. In this section we have proposed an algorithm to compute the performance parameter, mean number of packets and mean delay experienced by a packet for each TCP connection. To make the algorithm general we are considering that there are N TCP connections. So there will be N semaphore stations. Each semaphore station (S) has a input queue f(S) and a token queue e(S).

Assumptions:

- There are N TCP connections.
- There are M links in n^{th} TCP connection.
- Input queues of all the classes have finite buffer capacity. The buffer capacity of n^{th} class input queue is b_n
- The service discipline in the input queue of all the classes is of first come first serve (FCFS) type
- Packets arrive at the input queue of all the classes in Poisson fashion, at the rate of $\lambda_1, \lambda_2, \lambda_3 \ldots \lambda_N$ for the class 1, class 2, …,class N respectively.
- Arrival process at the token queue of n^{th} class $e_n(S)$ is described by state -dependent arrival rate $\gamma_n(k_1,k_2, \ldots,k_n,\ldots,k_N)$, where $k_1,k_2\ldots,k_n$ are the number of outstanding tokens for class 1,2…,N respectively. Here n varies from 1 to N.
- Interarrival times at the tokens queues of all the classes are exponentially distributed

Symbols Used:

C_n - Total number of tokens are permissions available for the n^{th} class TCP circuit.
k_n - Number of outstanding tokens for n^{th} class TCP circuit.
λ_n - Poisson arrival rate of packets at the input queue of n^{th} class TCP circuits.
$\gamma_n(k_1,k_2,\ldots,k_n)$ - state dependent arrival rate of the n^{th} class TCP connection when $k_1,k_2,\ldots,k_n$ number of tokens are out from the token queues of class 1,2….N respectively.
$q_n(j_1, j_2,\ldots, j_n)$ – joint probability distribution when there are $j_1, j_2, \ldots, j_n$ number of tokens are available in the token queue of class1,class2,. .,class n respectively.

$R_{m,n}(h_1,h_2,\ldots,h_n)$ – Avg. delay (including the processing delay at server) at the mth queue of nth class having $h_1,h_2,\ldots,h_n$ number of class1, class2,…,class n packets respectively in the N class closed queueing network (Q_N).

b_n - Buffer capacity of n^{th} class input queue.

Step0 Prepare the queueing model for the N TCP connections by using the N semaphore stations with N class of queueing networks as explained a case of two TCP connections in Section 3.3.

Step1: Draw the state diagram for the queueing model. Generally for N number of TCP connections there will be N dimensions and 2N tuple in the state of the system as shown in Fig 3 6 for the two TCP connections.

Step2: Obtain the state dependent arrival rates $\gamma_n(k_1,k_2,\ldots,k_N)$ of all the token queues by solving the N class closed queueing networks. The N class closed queueing networks can be obtained by linking the N queueing networks made up by the virtual routes of N TCP connections The state dependent arrival rate of particular class will be equal to the throughput of that particular class.

Step3: Evaluate the marginal probabilities of input and token queue of each class in a similar way as calculated in equations (3.17) to (3.20) for the case of two TCP connections

Step4: Compute the mean number of packets $L_{f_n(S)}$ at all the input queues $f_1(S)$, $f_2(S)$,…$f_N(S)$ of N semaphore stations by using the following relation.

For $n = 1,2,\ldots,N$

$$L_{f_n(S)} = \sum_{i=0}^{b_n} i p_n(i) \tag{3.21}$$

where $p_n(i)$ is the marginal probability distribution if there are i customers in queue $f_n(S)$.

Step5· Find out the mean delay experienced by the packet at each forward queue of the queueing networks of all classes and throughput of each class. These parameters can be found by solving the N class closed queueing network (Q_N) by using multiclass MVA algorithm described in Sec 3.5.

Algorithm 3.6 continued

Step6. Evaluate the mean number of packets, at each forward queue of the queueing network of all classes. The mean number of packets at the m th queue of n th class queueing network L_Q is given by following relation,

$$L_{Q_{nm}} = \sum_{h_1=0}^{C_1}\sum_{h_2=0}^{C_2} \cdot \sum_{h_n=0}^{C_n}[\gamma_n(h_1,h_2,\ldots,h_n)R_{nm}(h_1,h_2,\ldots,h_n)] * q_n(C_1-h_1,C_2-h_2,\ldots,C_n-h_n) \quad (3.22)$$

$n=1,2 \ldots N$ *and* $m=1,2\ldots M$. Assuming that there are M forward queues in the n th class queueing network

Step 7 Total mean number of packet for n th TCP connection will be,

$$L_{TCP} = L_{t_n(S)} + \sum_{n=1}^{N}\sum_{m=1}^{M} L_{Q_{nm}} \quad (3.23)$$

Step 8. Calculate the delay (D_n) for all the unshared links individually, in the n^{th} TCP connection. The delay for m^{th} unshared link in the n^{th} TCP connection is simply given by following equation.

$$D_{nm} = {L_{Q_{nm}}}/{\lambda_n} \quad (3.24)$$

Step 9: If m^{th} link of n^{th} class is shared with the l^{th} class then the mean delay (D_{nl}) provided by that link would be,

$$D_{nl} = \left(\frac{L_{nm} + L_{lm}}{\lambda_n + \lambda_l}\right) \quad (3.25)$$

Step 10: Compute the delay for n^{th} TCP connection T_n by adding the all link delay plus the delay provided by the input queue of n^{th} class semaphore station. Suppose there are m_1 unshared and m_2 shared links in the n^{th} TCP connections, then

$$T_n = \sum_{m=1}^{m_1} D_{nm} + \sum_{l=1}^{m_2} D_{nl} + \frac{L_{t_n(S)}}{\lambda_n} \quad (3.26)$$

Step 11: Utilization of n^{th} class semaphore queue is defined as percent of time the token queue of n^{th} class $e_n(S)$ is busy. First calculate the probability that token queue of n^{th} class is idle $p_{n(idle)}$,

$$p_{n(idle)} = \sum_{i_1=0}^{b_1} \sum_{j_1=0}^{C_1} \cdot\cdot \sum_{i_{n-1}=0}^{b_{n-1}} \sum_{j_{n-1}=0}^{C_{n-1}} p(i_1, j_1, \ldots, i_{n-1}, j_{n-1}, 0, C_n)$$

then the semaphore utilization of nth class $\rho_n(S)$ will be,

$$\rho_n(S) = 1 - p_{n(idle)} \tag{3.27}$$

On the basis of above modeling scheme presented in this chapter some examples have been solved and verified through the simulation in the next chapter.

Chapter 4

Examples and their validation through simulation

In Chapter 3 we have discussed that the multiple TCP connections can be modeled by using multiclass semaphore queues where each class represents the traffic of a particular TCP connection. Further in section 3.5 we have developed an approximate algorithm for solving these multiclass semaphore queues. By this approximate method, we can evaluate the performance parameters i.e , mean number of packets, utilization and mean delay experienced by a packet for each TCP connection. We also obtain the performance parameters through simulations and compare the results to show that our approximate approach is indeed applicable with some constraints. In this chapter we have given three examples to explain our algorithm properly For the simplicity of mathematical analysis, we have always considered that there are two TCP connections with one shared link. We have also assumed that the window size for all the classes are 3. In all the examples, dark lines (——) represent the class 1 traffic and the dashed lines (   ) represent the class 2 traffic.

Example 4.1

In this example we have considered two TCP circuits with three nodes as shown in Fig 4.1. The first TCP connection is from node 1 to node 2 (TCP_{12}) and the second TCP connection is from node 1 to node 3 (TCP_{13}). Link 1-2 (L_{12}) is shared by both the TCP connections For these connections, the basic queueing model is shown in Fig 4.2. In the model, input queues of both the classes have finite buffer capacity of 20 packets After accumulation of 20 packets in the input queue of any class, the arrival of packets of that particular class is blocked. The blocking is considered of rejection type. We have assumed that when a packet reaches its destination, it returns the token back to the token queue at higher service rate. The shared link L_{12} queue has the traffic of both the classes. The condition for the solution existence puts a limit over the maximum allowable arrival rates of different classes. According to that condition, the arrival rate of any class should be less than the maximum throughput achieved by that class in the two class closed queueing network obtained by linking Q1, Q2, Q3 and Q4. So, $\lambda_{12} < \gamma_1(3,0) = 1.99$ and $\lambda_{13} < \gamma_2(0,3) = 1.75$, where $\gamma_1(3,0)$ and $\gamma_2(0,3)$ are the maximum achieved throughputs for class 1 and class 2. Thus we have a common region of λ_{12} and λ_{13} under which we can operate our network without being congested. Choosing the arrival rates for both the classes within the maximum permissible value we have computed all the performance parameter (mean number of packets, mean delay experienced by a packet and semaphore utilization). Table 4 1 gives the service time for the different queues in the network. In Tables 4.2, 4.3, 4 4, 4 5 and 4.6 the computed and simulated values of performance parameters for TCP connection from node 1 to node 2 (TCP_{12}) are given. Confidence interval (for 96% confidence level) for the mean delay for TCP_{12} is shown in Table 4.6. Tables 4.7, 4.8, 4.9, 4.10 and 4.11 give the performance parameters for the TCP connection from node 1 to node 3 (TCP_{13}).

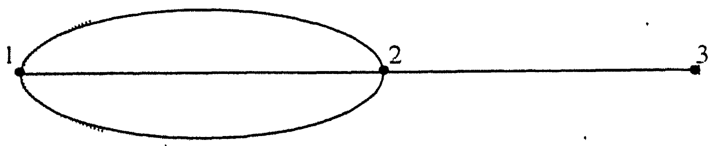

Fig. 4.1 An example of two TCP connections with one shared link (L_{12})

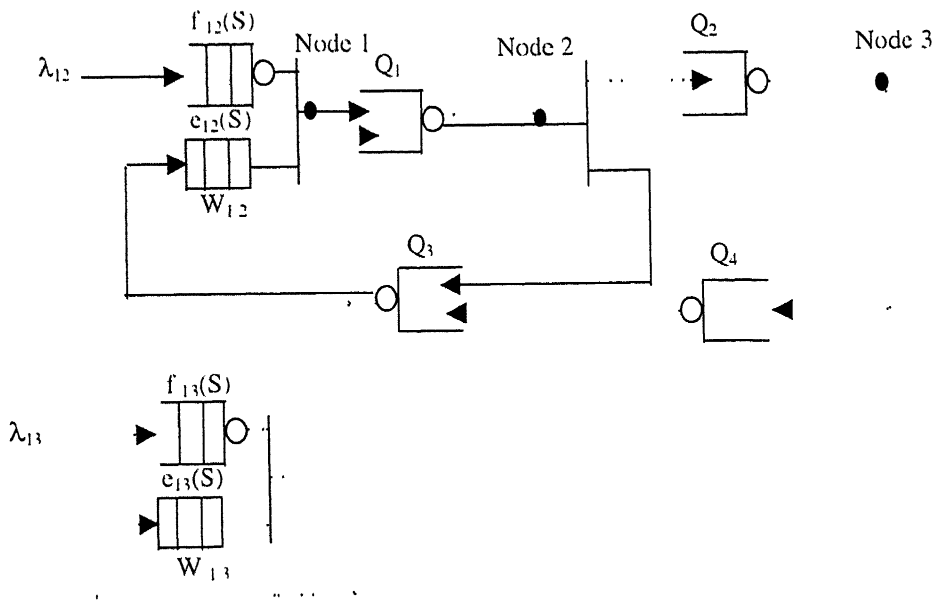

Fig 4.2 Queueing model of the TCP connections shown in Fig 4.1

Service time	Q_1	Q_2	Q_3	Q_4
Class 1	0.500000	0.000000	0.005000	0.000000
Class 2	0.500000	0.333333	0.003333	0.003333

Table 4.1 Service time of queues in Example 4.1

$\lambda^{13} \rightarrow$	0.0	0.4	0 6	0.8	1.0	1.3	1.5
$\lambda^{12} \downarrow$							
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.4	0 249995	0.658313	0 951001	1.312271	1.743263	2 475137	2.844466
0.8	0.666665	1 475727	2.168374	3 160460	4.401561	5.520821	5.645672
1.0	0.999994	2.292345	3.667474	6.234218	9.423144	10.57294	10 61654
1 3	1.856382	5.216269	9.683372	10.06901	19.55370	19.90349	19.90653
1.5	2.975883	9.532445	15.28298	19.70082	21.30012	21.43163	21.43232
1 7	5.170535	14.71102	18.85907	21.07132	21.75695	21.80682	21 80683

Table 4.2 Computed values of total mean number of packets for TCP_{12} in Example 4.1 on different arrival rates of the two classes

$\rightarrow \lambda^{13}$	0.0	0.4	0.6	0.8	1.0	1.3	1.5
$\downarrow \lambda^{12}$							
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.4	0.246597	0.667767	0.942049	1.321220	1.707395	2.559207	2.933003
0.8	0.661633	1.450518	2.082036	3.252737	4.147501	5.060212	5.617156
1.0	0.978032	2.297623	3.713592	6.575553	9.003660	9.515789	9.607699
1.3	1.823884	4.881128	9.176348	15.66753	19.48152	19.89205	19 91778
1.5	3.097229	9.437972	15.01001	20.10383	21 34211	21.98456	22.43388
1.7	4.924114	14.37081	18.73215	21.22953	23.02855	23.19526	23.40901

Table 4.3 Simulated values of total mean number of packets for TCP_{12} in Example 4.1 on different arrival rates of the two classes

$\rightarrow\lambda^{13}$ / $\downarrow\lambda^{12}$	0 0	0.4	0.6	0.8	1 0	1 3	1.5
0 0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0 4	0.624988	0.829742	0.964717	1.117782	1.284059	1 521856	1.577323
0 8	0.833331	1.288347	1.695575	2.305765	3.072908	3.595290	3.502446
1.0	0 999994	1.772591	2.692062	4.601787	7.074417	7.718342	7.577144
1 3	1.427986	3.517852	6.537328	11 03791	11.67354	11.83904	12.93449
1.5	1.983922	5.887124	9.416714	12.00989	12.15812	13.39333	13.48290
1.7	3.041491	8.244111	10.48541	10.48541	13.11046	13.41361	13.51893

Table 4.4 Computed values of total mean delay for TCP$_{12}$ in Example 4.1 on different values of arrival rates of the two classes

$\lambda^{13}\rightarrow$ / $\lambda^{12}\downarrow$	0.0	0.4	0.6	0.8	1.0	1.3	1.5
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.4	0.619248 ±0.01453	0.8379050 ±0.026661	0.956202 ±0.029610	1.115399 ±0.054414	1.259022 ±0.08333	1.570922 ±0.108986	1.645687 ±0.066141
0 8	0.824005 ±0.016457	1.257888 ±0.072959	1.622272 ±0.173435	2.341455 ±0.209988	2.806448 ±0.421977	3.273266 ±0.422163	3.767662 ±0.342017
1.0	0.976479 ±0.043214	1.769342 ±0.146928	2.699591 ±0.299600	4.808208 ±0.777572	6.547940 ±1.064917	6 959349 ±1.001051	7.045205 ±0.943901
1.3	1.402876 ±0.106476	3.240502 ±0.350234	6.131747 ±1.031055	10.73898 ±0.989932	13.00818 ±0.662464	13.28001 ±0.573151	13.47142 ±0.758148
1.5	2.056926 ±0.205610	5.854708 ±1.016200	9.368786 ±1.011263	12.56498 ±0.679899	13.10519 ±0.466659	13.26497 ±0 547242	13.65406 ±0.301698
1.7	2.900313 ±0 379893	8.144813 ±0.960387	10.61888 ±0.496163	12.07636 ±0.488890	13.78771 ±0.239284	13.80364 ±0.176766	13.92831 ±0.156895

Table 4.5 Simulated Values of total mean delay along with the confidence level (95%) for TCP$_{12}$ i Example 4.1 on different arrival rates of the two classes

$\rightarrow \lambda_{13}$ $\downarrow \lambda_{12}$	0.0	0.4	0.6	0.8	1.0	1.3	1.5
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.4	0.20	0.25	0.28	0.32	0.37	0.44	0.48
0.8	0.40	0.50	0.57	0.65	0.75	0.83	0.83
1.0	0.50	0.63	0.71	0.82	0.91	0.95	0.95
1.3	0.65	0.81	0.91	0.98	0.99	0.99	0.99
1.5	0.75	0.92	0.97	0.99	0.99	0.99	1.0
1.7	0.85	0.98	0.99	0.99	0.99	1.0	1.0

Table 4.6 Semaphore utilization for TCP_{12} circuit in Example 4.1 on different arrival rates of the two classes

$\rightarrow \lambda^{13}$ $\downarrow \lambda^{12}$	0.0	0.4	0.6	0.8	1.0	1.3	1.5
0.0	0.000000	0.405207	0.686635	1.062888	1.608835	3.219545	5.751063
0.4	0.000000	0.816724	1.252697	1.903909	3.032357	7.452738	13.06801
0.8	0.000000	1.514632	2.325374	3.924268	7.933891	17.59110	20.96825
1.0	0.000000	2.012965	3.164499	6.063170	13.80943	21.72857	23.25557
1.3	0.000000	2.956562	4.578696	8.728763	17.61804	21.99781	22.79039
1.5	0.000000	3.575430	4.974777	8.650791	17.16017	21.99780	22.79039
1.7	0.000000	3.903464	4.901616	8.286938	16.67282	21.49094	22.28232

Table 4.7 Computed value of total mean number of packets for TCP_{13} in Example 4.1 on different arrivals of the two classes

$\rightarrow \lambda^{14}$ $\downarrow \lambda^{13}$	0.0	0.4	0.6	0.8	1.0	1.3	1.5
0.0	0.000000	0.403134	0.678171	1.033525	1.559830	2.981866	5.302854
0.4	0.000000	0.823256	1 233418	1.887718	2.856874	7.700922	16.64139
0.8	0.000000	1.501842	2.243971	4.006852	7.407445	16.52192	20.43490
1.0	0.000000	2.002322	3.152712	5.971518	13.71441	20.95241	22.93688
1.3	0.000000	2.947438	4 501530	8.337294	17.86733	22 71049	23.63497
1.5	0.000000	3.577219	5.000023	8.512295	17.32968	22.95792	24.04288
1.7	0.000000	3.979959	5.100255	8.535374	17.74974	23.05455	23.99259

Table 4.8 Simulated values of total mean number of packets for TCP$_{13}$ in Example 4.1 on different arrival rates of the two classes

$\rightarrow \lambda^{14}$ $\downarrow \lambda^{13}$	0.0	0.4	0.6	0.8	1.0	1.3	1.5
0.0	0.000000	1.013017	1.144392	1.328610	1.608835	2.476573	3.834042
0.4	0.000000	1.225770	1.459923	1.839163	2.540501	5.291125	8.318487
0.8	0.000000	1.561330	2.071370	3.260525	6.379303	12.27974	12.96773
1.0	0.000000	1.783947	2.564686	5.028289	11.46070	14.16686	15.02517
1.3	0.000000	2.166532	3.352550	7.417596	14.16686	14.98424	15.59519
1.5	0.000000	1.153278	2.359582	3.466654	7.383100	14.70854	15.24768
1.7	0.000000	2.363387	3.287291	7.101326	13.58520	14.39461	14.96862

Tables 4.9 Computed values of total mean delay for TCP$_{13}$ in Example 4.1 on different arrival rates of the two classes

$\lambda \rightarrow$ $\lambda' \downarrow$	0.0	0.4	0.6	0.8	1.0	1.3	1.5
0.0	0.000000	1.008861 ±0.00296	1.132080 ±0.006675	1.294184 ±0.013043	1.558830 ±0.025814	2.295825 ±0.066396	3.538383 ±0.201995
0.4	0.000000	1.2248380 ±0.029409	1.4347250 ±0.039191	1.8087820 ±0.088760	2.3819110 ±0.226675	5.4528280 ±0 736528	9 7361870 ±0.916470
0.8	0.000000	1.544939 ±0.053055	1.989373 ±0.151717	3.250342 ±0.270955	5.851722 ±1.112740	11.66818 ±1 271792	12 99129 ±0.678777
1.0	0.000000	1.754194 ±0.079483	2.466975 ±0.156494	4.759171 ±0.475596	11.30665 ±1.243114	14.44471 ±0.720008	14.95308 ±0.306128
1.3	0.000000	2.131393 ±0.089538	3.121519 ±0.251677	6.731798 ±0.821575	14.93278 ±1.254545	15 15981 ±0.423810	16.09985 ±0 214565
1.5	0.000000	2.378580 ±0.130357	3.394313 ±0.188566	6.785098 ±0.821575	14.50786 ±1.254545	15.24786 ±0.423810	16.31202 ±0.214565
1.7	0.000000	2.503453 ±0.087056	3.374065 ±0.110528	6.790246 ±0.891070	15.09759 ±1.012969	15.28152 ±0.311935	16.41082 ±0.173812

Table 4.10 Simulated values of total mean delay along with the confidence level (96%) for TCP_{13} in Example 4.1 on different arrival rates of the two classes

$\rightarrow \lambda_1$ $\downarrow \lambda_1'$	0.0	0.4	0.6	0.8	1.0	1.3	1.5
0.0	0.00	0.31	0.44	0.57	0.68	0.82	0 90
0.4	0.00	0.35	0.50	0.64	0.76	0.91	0.97
0.8	0.00	0.42	0.60	0.76	0.89	0.99	0.99
1.0	0.00	0.46	0.66	0.84	0.96	0.99	0.99
1.3	0.00	0.55	0.77	0.93	0.99	0.99	1.0
1.5	0.00	0.60	0.81	0.94	0.99	0.99	1.0
1.7	0.00	0.64	0.83	0.95	0.99	0.99	1.0

Table 4.11 Semaphore utilization for TCP_{13} circuit in Example 4.1 on different arrival rates of the two classes

It can be seen that the results are close to the results obtained from simulations and are within the confidence intervals for a 96% confidence level. It can be noted that in case of high semaphore utilization the approximate average delay is slightly overestimated. The relative error is within limit for utilization upto 0.85. Thus it can be concluded that accuracy of approximate algorithm depends upon the semaphore utilization for each TCP circuit.

Example 4.2

This example also considers two TCP connections with one shared link in a three node network as shown in Fig 4.3. The first TCP connection is from node 1 to node 3 (TCP_{13}) and the second TCP connection is from node 2 to node 3 (TCP_{23}). Link from node 2 to node 3 (L_{23}) is shared between both the TCP circuits. The buffer capacity of the input queue is 20 According to the condition of solution existence, $\lambda_{13} < \gamma_1(3,0) = 1.573$ and $\lambda_{23} < \gamma_2(0,3) = 4.0$, where $\gamma_1(3,0)$ and $\gamma_2(0,3)$ are the maximum achieved throughputs for class 1 and class 2. Choosing the arrival rates for both the classes within the above maximum permissible value we have computed the performance parameters (total mean number of packets, total mean delay experienced by a packet and semaphore utilization) for each TCP connection. Table 4.12 gives the service time for the different queues in the network. In Tables 4.13, 4.14, 4.15, 4.16, and 4.17 the computed and simulated values of performance parameters for TCP connection from node 1 to node 3 (TCP_{13}) are given. Tables 4.18, 4.19, 4.20, 4.21, and 4.22 give the performance parameters for the TCP connection from node 2 to node 3 (TCP_{23}).

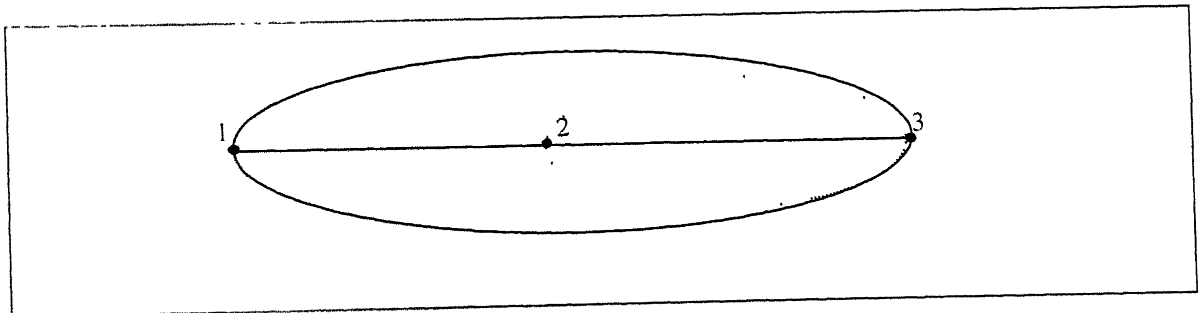

Fig. 4.3 An example of two TCP connections with shared link L_{23}

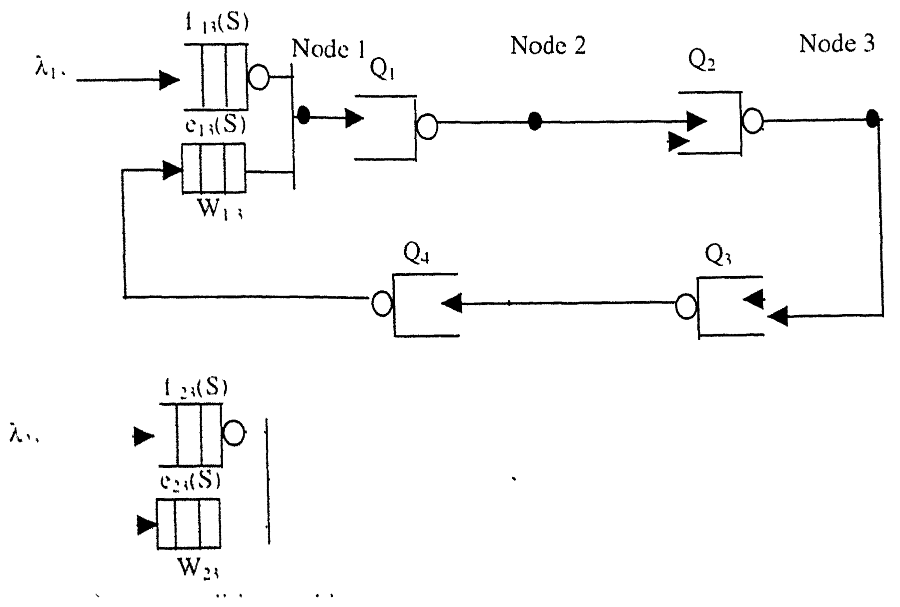

Fig 4.4 Queueing model for the Example 4.2

Service time	Q_1	Q_2	Q_3	Q_4
Class 1	0.500000	0.333333	0.003333	0.050000
Class 2	0.000000	0.250000	0.002500	0.000000

Table 4.12 Service time of all queues in Example 4.2

$\rightarrow \lambda_{13}$ $\downarrow \lambda_{23}$	0.0	0.5	0.7	0.9	1.1	1.3	1.5
0.0	0.000000	0.536764	0.859422	1.307039	1.993890	3.219232	5.750450
0.5	0.000000	0.742212	1.107094	1.623669	2.446258	4.001900	7.291587
1.0	0.000000	1.028773	1.464818	2.102774	3.182603	5.400415	9.913705
1.5	0.000000	1.425323	1.981947	2.845975	4.475437	8.077865	13 86991
2.0	0.000000	1.961866	2.717234	4.026743	6.972412	12.86821	18.17732
3.0	0.000000	3.440868	4.610153	6.844387	12.43600	18.84812	21.29049
3.5	0.000000	3.987033	5.016369	7.116753	12.61399	18.92068	21.39605

Tables 4.13 Computed values of total mean number of packets for TCP_{13} in Example 4.2 on different arrival rates of two classes.

$\rightarrow \lambda_{13}$	0.0	0.5	0.7	0.9	1.1	1.3	1.5
$\downarrow \lambda_{23}$							
0.0	0.000000	0.538367	0.831310	1.241744	1.963023	2.892759	5.230676
0 5	0 000000	0.739534	1.117602	1.602262	2.308375	3.596869	6.559357
1 0	0.000000	1.053859	1.480007	1.998221	2.955263	4.640457	8.870600
1 5	0 000000	1.427024	1.980505	2.767076	4.116513	7.270308	12.47562
2 0	0.000000	1.883016	2.579105	3.817633	5.291739	9.777785	16.85057
3.0	0.000000	3.299112	4.193589	5.331075	7.621608	14.54683	20.66421
3.5	0 000000	3.742181	4.596806	5.483952	7.666961	14.73301	20.21921

Table 4.14 Simulated value of Mean number of packets for TCP$_{13}$ in Example 4.2 on different arrival rates of two classes

$\rightarrow \lambda_{13}$	0.0	0.5	0.7	0.9	1.1	1.3	1.5
$\downarrow \lambda_{23}$							
0.0	0.000000	1.073528	1.227746	1.452266	1.812628	2.476633	3.833634
0.5	0.000000	1.083092	1.263974	1.532442	1 981032	2.855451	4.654967
1.0	0.000000	1.145428	1.360073	1.699498	2.314675	3 616432	6.115294
1.5	0.000000	1.240021	1.514973	1.995903	2.990594	5.209728	8 366225
2.0	0.000000	1.366627	1.747840	2.537410	4.515658	8.267868	10.82145
3.0	0.000000	1.673312	2.310475	3.910584	8.165964	12 11820	12.46131
3.5	0.000000	1.728677	2.340284	3.927561	8.165505	12.06104	12.39019

Table 4.15 Computed Value of total mean delay of packets for TCP$_{13}$ in Example 4.2 on different arrival rates of two classes

$\lambda_{13}\rightarrow$ $\lambda_{23}\downarrow$	0.0	0.5	0.7	0.9	1.1	1.3	1.5
0.0	0.000000	1.070545 ±0.010343	1.193006 ±0.023136	1.383247 ±0.042309	1.780185 ±0.115312	2.239742 ±0.139437	3.465634 ±0.520762
0.5	0.000000	1.074604 ±0.015753	1.259552 ±0.036903	1.494504 ±0.070909	1.841342 ±0.091024	2.524223 ±0.204575	4.118170 ±0.782437
1.0	0.000000	1.165666 ±0.034541	1.343476 ±0.047995	1.588876 ±0.081484	2.082701 ±0.178883	3.018454 ±0.541194	5.360453 ±0.973615
1.5	0.000000	1.217913 ±0.040448	1.455818 ±0.084925	1.861365 ±0.141982	2.599897 ±0.246742	4.441034 ±0.795726	7.629502 ±1.137693
2.0	0.000000	1.297906 ±0.067362	1.592525 ±0.116451	2.213591 ±0.214894	3.026420 ±0.344178	5.941083 ±1.591557	10.47425 ±1.089477
3.0	0.000000	1.570963 ±0.110966	1.941121 ±0.143362	2.554221 ±0.234129	4.091444 ±0.409603	9.154947 ±1.159090	13.47244 ±0.529901
3.5	0.000000	1.623624 ±0.084310	2.073842 ±0.117079	2.624049 ±0.219907	4.065880 ±0.622420	9.213251 ±1.530425	13.05343 ±0.901044

Table 4.16 Simulated values of total mean delay along with the confidence level (96%) for TCP_{13} in Example 4.2 on different arrival rates of the two classes

$\rightarrow\lambda_{13}$ $\downarrow\lambda_{23}$	0.0	0.5	0.7	0.9	1.1	1.3	1.5
0.0	0.00	0.38	0.51	0.62	0.73	0.82	0.90
0.5	0.00	0.40	0.53	0.65	0.75	0.84	0.92
1.0	0.00	0.42	0.56	0.68	0.79	0.88	0.95
1.5	0.00	0.45	0.60	0.72	0.83	0.93	0.98
2.0	0.00	0.49	0.65	0.78	0.89	0.97	0.99
3.0	0.00	0.60	0.76	0.89	0.97	0.99	1.0
3.5	0.00	0.63	0.79	0.90	0.97	0.99	1.0

Table 4.17 Semaphore utilization for TCP_{13} circuit in Example 4.2 on different arrival rates of the two classes

$\rightarrow \lambda_{13}$ $\downarrow \lambda_{23}$	0.0	0.5	0.7	0.9	1.1	1.3	1.5
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.5	0.142855	0.401934	0.534378	0.685626	0.856232	1.045161	1.238796
1.0	0.333336	0.696271	0.888764	1.112563	1.367884	1.648247	1.901100
1.5	0.600005	1.155038	1.471208	1.852724	2.297799	2.756942	3.019532
2.0	0.999985	1.967535	2.612691	3.493752	4.617132	5.542602	5.670956
3.0	2.975886	7.899132	11.53804	15.66764	19.08482	19.94986	19.55204
3.5	5.984297	14.20426	17.46913	20.09153	21.83927	22.01896	21.53363

Table 4.18 Computed values of mean number of packets for TCP_{23} in Example 4.2 on different arrival rates of the two classes

$\rightarrow \lambda_{13}$ $\downarrow \lambda_{23}$	0.0	0.5	0.7	0.9	1.1	1.3	1.5
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.5	0.142556	0.403137	0.556938	0.708999	0.880174	1.082139	1.270977
1.0	0.334122	0.718332	0.933657	1.143292	1.430323	1.688572	2.089139
1.5	0.598376	1.186264	1.547084	2.017595	2.523083	3.338960	3.457331
2.0	0.984175	1.932490	2.714648	4.025331	4.762155	7.034862	7.709817
3.0	3.169375	8.250277	10.78437	14.38448	17.79201	19.95772	20.33832
3.5	5.723566	13.40011	17.09567	18.36855	20.45473	21.71151	22.03756

Table 4.19 Simulated value of total mean number of packets for TCP_{23} in Example 4.2 on different arrival rates of the two classes

$\rightarrow \lambda_{13}$ / $\downarrow \lambda_{23}$	0.0	0.5	0.7	0.9	1.1	1.3	1.5
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.5	0.285709	0.402537	0.446282	0.491155	0.537126	0.583293	0.622770
1.0	0.333360	0.468242	0.529827	0.596659	0.667799	0.739478	0.789952
1.5	0.400003	0.591067	0.694127	0.815284	0.952150	1.083834	1.132635
2.0	0.499993	0.823949	1.044940	1.354685	1.757139	2.082305	2.106050
3.0	0.991962	2.488421	3.613242	4.890063	5.939521	6.202975	6.084251
3.5	1.709799	3.930905	4.798143	5.477277	5.913665	5.947156	5.818087

Table 4.20 Computed values of total mean delay of packets for TCP_{23} in Example 4.2 on different arrival rates of the two classes

$\lambda_{13} \rightarrow$ / $\lambda_{23} \downarrow$	0.0	0.5	0.7	0.9	1.1	1.3	1.5
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.5	0.284529 ±0.002572	0.402589 ±0.009651	0.463232 ±0.015797	0.508205 ±0.018383	0.553130 ±0.029043	0.604675 ±0.036038	0.645497 ±0.033963
1.0	0.334080 ±0.006545	0.482925 ±0.023157	0.556311 ±0.035205	0.621838 ±0.036737	0.700099 ±0.066450	0.777576 ±0.075455	0.892925 ±0.096874
1.5	0.397921 ±0.032280	0.608769 ±0.050223	0.731692 ±0.083294	0.913737 ±0.146477	1.074725 ±0.117771	1.372632 ±0.267642	1.373321 ±0.145064
2.0	0.491277 ±0.042630	0.813355 ±0.129215	1.101382 ±0.233361	1.593406 ±0.435681	1.849072 ±0.568262	2.812897 ±1.017156	3.054440 ±0.699525
3.0	1.046695 ±0.258661	2.651531 ±0.812021	3.516489 ±0.905407	4.851748 ±0.922943	6.408790 ±0.722094	7.714663 ±0.552360	7.889542 ±0.612332
3.5	1.643253 ±0.315946	3.970105 ±0.899291	5.412559 ±0.635513	6.145913 ±0.679030	7.332066 ±0.459294	8.365044 ±0.259774	8.562465 ±0.288654

Table 4.21 Simulated values of total mean delay of packets along with confidence level (96%) for TCP_{23} in Example 4.2 on different arrival rates of the two classes

$\rightarrow \lambda_{13}$ / $\downarrow \lambda_{23}$	0.0	0.5	0.7	0.9	1 1	1.3	1.5
0.0	0 00	0 00	0.00	0.00	0.00	0.00	0.00
0 5	0 13	0 15	0 16	0.18	0.19	0.20	0.23
1 0	0 25	0.30	0.33	0.35	0.38	0.42	0 44
1.5	0 38	0.45	0.49	0.53	0.58	0.63	0 66
2 0	0 50	0.60	0.65	0.71	0.77	0.83	0.85
3.0	0.75	0.89	0.94	0.97	0 99	0.99	0.99
3 5	0 87	0.97	0.99	1.00	1.00	1.00	1.0

Table 4.22 Semaphore utilization for TCP_{23} circuit in Example 4.2 on different arrival rates of the two classes

Again the results are close to the results obtained from simulations and are also well within the confidence intervals of 96%. It can also be noted that in the case of high semaphore utilization the approximate average delay is slightly overestimated. Again the relative error is within limit for utilization upto 0.85.

Example 4.3

This example considers two TCP connections with one shared link in four nodes as shown in Fig 4.5. The first TCP connection is from node 1 to node 3 (TCP_{13}) and the second TCP connection is from node 2 to node 4 (TCP_{24}). Link from node 2 to node 3 (L_{23}) is shared between both the TCP circuits The buffer capacity of the input queue is 20. According to the condition of solution existence, $\lambda_{13} < \gamma_1(3,0) = 1.7517$ and $\lambda_{24} < \gamma_2(0,3) = 0.9$, where $\gamma_1(3,0)$ and $\gamma_2(0,3)$ are the maximum achieved throughputs for class 1 and class 2. Choosing the arrival rates for both the classes within the above maximum permissible value we have computed all the performance parameter (mean number of packets, mean delay experienced by a packet and semaphore utilization) Table 4.23 gives the service time for the different queues in the network. In Tables 4.24, 4.25, 4.26, 4.27, and 4.28 the computed and simulated values of performance parameters for TCP connection from node 1 to node 3 (TCP_{13}) are given. Tables 4.29, 4.30, 4.31, 4.32, and 4.33 give the performance parameters for the TCP connection from node 2 to node 4 (TCP_{24})

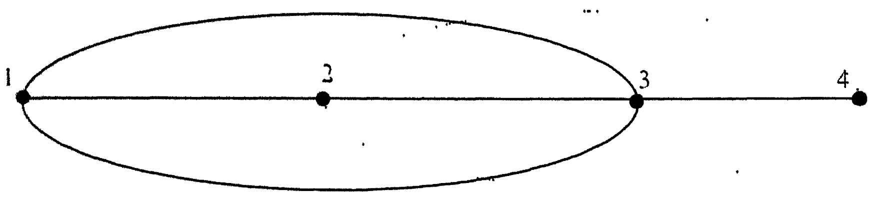

Fig. 4.5 An example of two TCP connections with shared link L_{23}

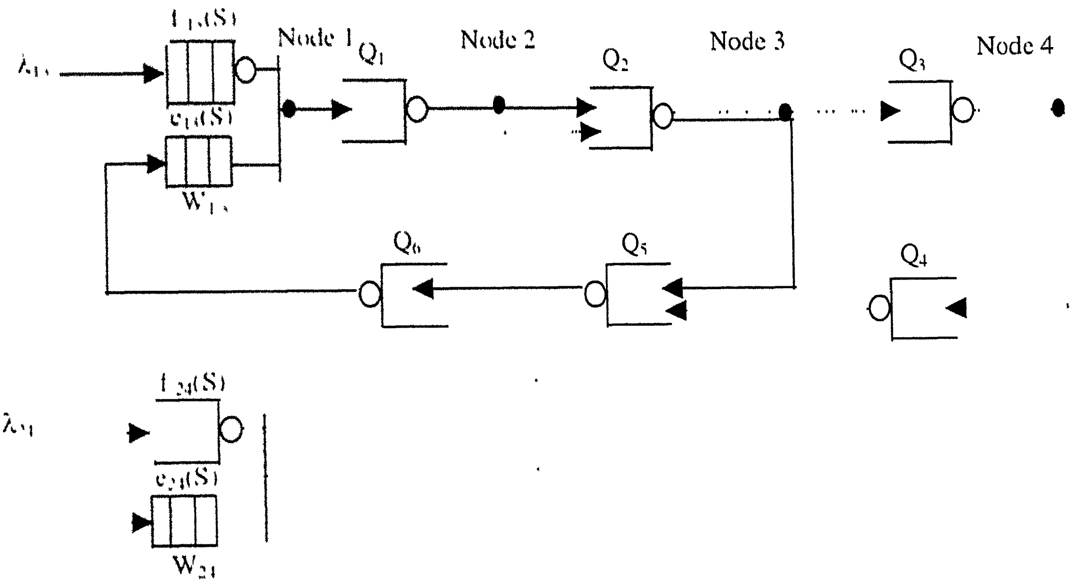

Fig 4.6 Queueing model of the TCP connections shown in Example 4.3

Service time	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6
Class 1	1.000000	0.500000	0.000000	0.000000	0.0500000	0.010000
Class 2	0.000000	0.500000	0.333333	0.030000	0.0500000	0.000000

Table 4.23 Service time of the queues in Example 4.3

$\lambda_{21} \rightarrow$ / $\lambda_{13} \downarrow$	0.0	0.4	0.6	0.8	1.2	1.5
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.2	0.357075	0.667582	0.882117	1.153299	1.837116	2.353156
0.3	0.599083	0.962437	1.221923	1.510072	2.293596	2.814021
0.5	1.399435	1.903748	2.261320	2.646420	3.790343	4.322005
0.6	2.020609	2.709428	3.257034	3.599365	5.090328	5.964949
0.8	5.963893	7.189785	7.782830	10.326239	13.542427	15.299444

Table 4.24 Computed values of total mean number of packets for TCP_{13} in Examples 4.3 on different arrival rates of two classes

$\lambda_{21} \rightarrow$ / $\lambda_{13} \downarrow$	0.0	0.4	0.6	0.8	1.2	1.5
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.2	0.344635	0.664472	0.895196	1.156328	1.815995	2.369947
0.3	0.546815	0.974014	1.228131	1.536620	2.3120631	2.924948
0.5	1.382889	1.908382	2.294013	2.784584	4.087705	4.798449
0.6	2.075835	2.771509	3.314634	4.048960	6.190749	7.091616
0.8	5.379027	7.606651	9.188235	11.333368	16.356495	17.336520

Table 4.25 Simulated value of total mean number of packets for TCP_{13} in Example 4.3 on different arrival rates of the two class

$\lambda_{21} \rightarrow$ / $\lambda_{1\cdot} \downarrow$	0.0	0.4	0.6	0.8	1.2	1.5
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.2	1.723173	1.970896	2.067001	2.171512	2.394351	2.531101
0.3	1.822715	2.228878	2.350200	2.486883	2.785450	2.945083
0.5	2.765777	3.114868	3.364486	3.680606	4.513107	4.842981
0.6	3.459724	3.996448	4.428056	3.955673	6.953643	7.627365
0.8	7.516900	8.992292	10.571195	12.747743	17.893466	18.732163

Table 4.26 Computed values of mean delay of packets for TCP_{13} in Example 4.3 on different arrival rates of the two classes

$\lambda_{2\cdot} \rightarrow$ / $\lambda_{1\cdot} \downarrow$	0.0	0.4	0.6	0.8	1.2	1.5
0.0	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
0.2	1.790863 ±0.008371	1.944839 ±0.014729	2.022885 ±0.028703	2.142783 ±0.033513	2.426001 ±0.074087	2.474696 ±0.072944
0.3	1.994682 ±0.014001	2.210588 ±0.265700	2.320038 ±0.046322	2.419763 ±0.060955	2.710713 ±0.077244	2.824404 ±0.077081
0.5	2.766096 ±0.068170	3.050388 ±0.091639	3.264342 ±0.102472	3.416973 ±0.124023	4.036520 ±0.237410	4.230179 ±0.212995
0.6	3.355461 ±0.125897	3.840726 ±0.172511	4.246912 ±0.200693	4.373462 ±0.245094	5.267864 ±0.415882	6.037222 ±0.506638
0.8	7.406536 ±0.612393	8.544214 ±1.001818	9.020261 ±1.057076	11.772095 ±1.102763	15.059066 ±1.537379	17.159313 ±1.289069

Table 4.27 Simulated values of total mean delay along with the confidence level (95%) for TCP_{13} in Example 4.3 on different arrival rates of the two class

$\lambda_{24} \rightarrow$ $\lambda_{13} \downarrow$	0.0	0.4	0.6	0.8	1.2	1.5
0.0	0.00	0.00	0.00	0.00	0.00	0.00
0.2	0.28	0.30	0.31	0.33	0.37	0.40
0.3	0.40	0.43	0.45	0.47	0.52	0.56
0.5	0.63	0.66	0.68	0.71	0.78	0.81
0.6	0.73	0.76	0.78	0.81	0.88	0.90
0.8	0.90	0.92	0.94	0.96	0.99	0.99

Table 4.28 Semaphore utilization for TCP_{13} in Example 4.3 on different arrival rates of the two classes

$\lambda_{24} \rightarrow$ $\lambda_{13} \downarrow$	0.0	0.4	0.6	0.8	1.2	1.5
0.0	0.000000	0.404107	0.691980	1.022624	2.338044	5.230676
0.2	0.000000	0.574966	0.918958	1.372668	3.455853	8.385462
0.3	0.000000	0.693329	1.071749	1.593097	4.048441	10.19641
0.5	0.000000	0.962875	1.465194	2.116990	6.020378	13.60135
0.6	0.000000	1.122664	1.689210	2.448498	7.731296	16.14395
0.8	0.000000	1.398324	2.028319	3.187247	10.230007	18.777145

Table 4.29 Computed values of total mean number of packets for TCP_{24} in Example 4.3 on different arrival rates of the two classes

$\lambda_{24} \rightarrow$ / $\lambda_{13} \downarrow$	0.0	0.4	0.6	0.8	1.2	1.5
0.0	0.000000	0.405075	0.686415	1.062466	2.504634	5.743933
0.2	0.000000	0.582944	0.929593	1.407493	3.507057	8.730093
0.3	0.000000	0.697007	1.080444	1.629630	4.275633	10.778099
0.5	0.000000	0.955069	1.442766	2.191697	6.755826	15.439195
0.6	0.000000	1.101962	1.653308	2.535262	8.649748	17.658878
0.8	0.000000	1.411533	2.079846	3.184258	11.731392	19.799150

Table 4.30 Simulated value of total mean number of packets for TCP_{24} in Example 4.3 on different arrival rates of the two classes

$\lambda_{24} \rightarrow$ / $\lambda_{13} \downarrow$	0.0	0.4	0.6	0.8	1.2	1.5
0.0	0.000000	1.012687	1.144025	1.328083	2.087195	3.829289
0.2	0.000000	1.111674	1.281657	1.533733	2.736836	5.654397
0.3	0.000000	1.168941	1.365003	1.666465	3.253653	6.913210
0.5	0.000000	1.290959	1.554929	2.001902	4.994038	9.764584
0.6	0.000000	1.353755	1.659180	2.102218	6.367056	11.101866
0.8	0.000000	1.464745	1.841345	2.561356	8.641870	12.363597

Table 4.31 Computed values of total mean delay of packets for TCP_{24} in Examples 4.3 on different arrival rates of the two classes

λ_2: → / λ_1: ↓	0.0	0.4	0.6	0.8	1.2	1.5
0.0	0.000000	1.013578 ±0.008266	1.146941 ±0.099960	1.282298 ±0.029636	1.955302 ±0.147983	3.471828 ±0.520762
0.2	0.000000	1.090972 ±0.011570	1.270066 ±0.025754	1.499092 ±0.039073	2.684030 ±0.045873	5.487391 ±0.263698
0.3	0.000000	1.159930 ±0.017213	1.360038 ±0.655513	1.624172 ±0.071904	3.036500 ±0.294078	6.633629 ±1.046059
0.5	0.000000	1.293772 ±0.022732	1.563882 ±0.056481	1.921654 ±0.118997	4.410087 ±0.643488	9.182566 ±0.890365
0.6	0.000000	1.367645 ±0.042356	1.676032 ±0.070926	2.118033 ±0.153611	5.752057 ±0.955775	11.31798 ±0.860876
0.8	0.000000	1.459917 ±0.043733	1.813496 ±0.086004	2.553825 ±0.253025	7.538430 ±1.019470	13.782632 ±0.897895

Table 4.32 Simulated values of total mean delay for TCP_{24} in Example 4.3 on different arrival rates of the two classes

λ_2: → / λ_1: ↓	0.0	0.4	0.6	0.8	1.2	1.5
0.0	0.000000	0.31	0.44	0.56	0.78	0.90
0.2	0.000000	0.33	0.47	0.60	0.81	0.94
0.3	0.000000	0.34	0.48	0.61	0.84	0.96
0.5	0.000000	0.35	0.52	0.66	0.89	0.99
0.6	0.000000	0.37	0.54	0.68	0.92	1.0
0.8	0.000000	0.41	0.58	0.74	0.96	1.0

Table 4.33 Semaphore utilization for TCP_{24} circuit in Example 4.3 on different arrival rates of the two classes

In this example also, the results are close to the results obtained from simulations and are well within the confidence intervals of 96%. It can be noted again that in the case of high semaphore utilization the approximate average delay is slightly overestimated. The relative error is within limit for utilization upto 0.85.

Chapter 5

Conclusions and Suggestions for Future work

Transmission control protocol is used when we need sustained interchange of data between two machines. To provide the reliability of data transfer it uses the sliding window mechanism These TCP connections with sliding window principle have been modeled by using semaphore queues. The desciiption of semaphore queues is described in the earliei part of this thesis. Basically semaphore queues can be considered as the means of controlling the number of customers in a queueing network. It permits us to model node to node and end to end window flow controls. We have modeled the single TCP connection by using a semaphore queue and replacing all the links by a infinite buffer queue.

In this thesis multiple TCP connections with shared links are emphasize. Multiple TCP connections are modeled with the help of multiple semaphore queues. In order to differentiate the traffic of one TCP connection from another one a separate class for each TCP connection is used. Thus we have modeled the multiple TCP connections by using multiple semaphore queues with multiclass traffic where each class represents the traffic of a particular TCP connection. First we have given an approach to solve the two class semaphore queues for two TCP connections. Further we have derived an approximate method to solve this multiclass semaphore queues in the same way as in the case of two TCP connections. By this method we can evaluate the performance of each TCP connection. Performance parameters include total mean number of packets, total mean delay experienced by a packet and utilization of the semaphore queues. Finally we have validated our computed results through simulations. The theoretical results are close to the iesults obtained from simulations and are also well within the confidence intervals for a 96% confidence level. We also note that in the case of high semaphore utilization the approximate average delay is slightly overestimated. The relative error is within limit for utilization up to 0.85. Thus finally, it is concluded that accuracy of approximate algorithm given in this thesis depends upon the semaphore utilization for each TCP circuit

Suggestions for the Future work

We have presented an approximate algorithm to evaluate the performance of multiple TCP circuits with partially shared link The solution existence criteria puts a restriction over the arrival rates of different classes that we can permit to go inside the network We have also seen that our algorithm works properly below the semaphore utilization 0.85 Thus we have a safer region between the arrival rates of different class where we can operate our network without being congested. In future, the work can be extended to find the analysis algorithm for semaphore utilization greater than 0 85. This model can be incorporated in traffic flow management software in a network.

QNAT (Queueing network Analysis Tool) is a software package developed in EE/ACES, IIT Kanpur and is used to analyze queueing networks. Since in the algorithm presented in this thesis we need to solve multiclass closed queueing networks it can be incorporated in QNAT

Bibliography

[1]. Serge Fdida, H.G.Perros and Andrzej Wilk, "*Semaphore queues: Modelling Multilayered Window Flow Control Mechanism*", IEEE Transaction on communication vol.38 No.3 March 1990.

[2]. Doughlas E. Comer, "*Internetworking with TCP/IP*", volume I, Third Edition PHI publications.

[3]. M. Reiser," *A queueing network analysis of computer communication networks with window flow control*", IEEE Trans. Comm Vol. COM-27, pp.1199-1209, 1979.

[4]. C. Kim and Agrawala, "Analysis of Fork-Join queue", IEEE Trans. On computers, Vol. 38, NO.2, pp.250-255, Feb.1989.

[5]. Leonard Kleinrock, "*Queueing systems*", Volume I: Theory, Wiley-Interscience publication.

[6]. S.S.Lavenberg, "*Stability and maximum departure rate of certain open queueing networks having capacity constraints*", RAIRO Informatique/computer Science, Vol. 12, pp.353-370, 1978.

[7]. F.Baskett, K.M. Chandy, R.R. Muntz and F.G. Palacios,"open, closed and mixed networks of queues with different class of customers", J of ACM, Vol.22, No. 2,pp.248-260, April 1975.

[8]. M. Reiser and S.S.Lavenberg," Mean Value Analysis of closed multichain queueing networks", J. of ACM, Vol.27, No.2, pp.313-322, April 1980.

[9]. Michael K. Molloy," Fundamental of performance modelling ", Macmillan publishing company, 1989.

TH
EE/1999/M
K96 þ